

--NTSHEMBO MALUEKE

--Practical 3

Q1

The screenshot shows a database interface with multiple tabs at the top: Practical 1, Practical 2, Practical 3, Bright\_Coffee\_Case\_study 1, BrightTV\_Case\_study 2, and a '+' tab. The Practical 3 tab is active. The code in the editor is:

```
1 --NTSHEMBO MALUEKE
2 --Practical 3
3
4 -----
5 --1. Find all records where Size is missing and the purchase_amount is greater than 50.
6 --Expected Columns: Customer ID, Size, purchase_amount, Item Purchased
7
8 select customer_id,
9      size,
10     purchase_amount,
11     item_purchased
12    from shopping_tread
13   where size is null
14      and purchase_amount > 50;
```

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

CUSTOMER_ID	SIZE	PURCHASE_AMOUNT	ITEM_PURCHASED
1	11	74.0	Handbag
2	15	54.0	Jeans
3	22	88.0	Shirt
4	32	54.0	Blouse
5	A2	57.0	Riviera

Q2

The screenshot shows a database interface with multiple tabs at the top: Practical 1, Practical 2, Practical 3, Bright\_Coffee\_Case\_study 1, BrightTV\_Case\_study 2, and a '+' tab. The Practical 3 tab is active. The code in the editor is:

```
15
16 -----
17 --2. List the total number of purchases grouped by Season, treating NULL values as 'Unknown Season'.
18 --Expected Columns: Season, Total Purchases
19
20 select
21     coalesce(season, 'unknown Season') as season,
22     count(item_purchased) as total_purchase
23    from shopping_tread
24   group by season;
```

The results table has two columns: SEASON and TOTAL\_PURCHASE. The data is:

SEASON	TOTAL_PURCHASE
Summer	58
Winter	71
unknown Season	26
Fall	50
Spring	66

### Q3

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

```
27
28
29
30     group by season;
31
--3. Count how many customers used each Payment Method, treating NULLs as
--"Not Provided".
--Expected Columns: Payment Method, Customer Count
32
33
34     select coalesce(payment_method, 'Not Provided') as PAYMENT_METHOD,
35         count(distinct customer_id) as customer_count
36     from shopping_tread
37     group by PAYMENT_METHOD;
```

The results table has two columns: PAYMENT\_METHOD and CUSTOMER\_COUNT. The data is:

PAYMENT_METHOD	CUSTOMER_COUNT
PayPal	51
Bank Transfer	38
Debit Card	42
Venmo	53
Not Provided	30
Cash	42

### Q4

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

```
38
39
40     --4. Show customers where Promo Code Used is NULL and Review Rating is below 3.0.
41     --Expected Columns: Customer ID, Promo Code Used, Review Rating, Item Purchased
42
43     select customer_id,
44         promo_code_used,
45         review_rating,
46         item_purchased
47     from shopping_tread
48     where promo_code_used is null
49     and review_rating < 3.0;
```

The results table has four columns: CUSTOMER\_ID, PROMO\_CODE\_USED, REVIEW\_RATING, and ITEM\_PURCHASED. The data is:

CUSTOMER_ID	PROMO_CODE_USED	REVIEW_RATING	ITEM_PURCHASED
61	null	2.5	Jeans
80	null	2.6	Sneakers
125	null	2.8	Sneakers
128	null	2.5	Shoes
180	null	2.5	Shorts
285	null	2.9	Blouse

On the right side, there is a 'Query Details' panel showing:

- Query duration: 609ms
- Rows: 8
- Query ID: 01c02b94-000c-b2eb-...

## Q5

The screenshot shows a data analysis interface with multiple tabs at the top: Practical 1, Practical 2, Practical 3, Bright\_Coffee\_Case\_study 1, BrightTV\_Case\_study 2, and a new tab labeled 'Bright TV Case Study 3'. The current tab is 'Bright TV Case Study 3'. The interface includes a toolbar with 'Share' and 'Code Versions' buttons, and a search bar.

The code editor displays the following SQL query:

```
50
51
52
53
54
55
56
57
58
59
60
```

--5. Group customers by Shipping Type, and return the average purchase\_amount, treating missing values as 0.  
--Expected Columns: Shipping Type, Average purchase\_amount

```
select shipping_type,
       avg(coalesce(purchase_amount, 0)) as Average_purchase_amount
  from shopping_tread
 group by shipping_type;
```

The results table shows the average purchase amount for each shipping type:

SHIPPING_TYPE	AVERAGE_PURCHASE_AMOUNT
Standard	47.6666667
Express	53.4545455
Store Pickup	55.3333333
null	52.7037037
Free Shipping	50.2142857
Next Day Air	54.8666667
2-Day Shipping	51.5576923

## Q6

The screenshot shows a data analysis interface with multiple tabs at the top: Practical 1, Practical 2, Practical 3, Bright\_Coffee\_Case\_study 1, BrightTV\_Case\_study 2, and a new tab labeled 'Bright TV Case Study 3'. The current tab is 'Bright TV Case Study 3'. The interface includes a toolbar with 'Share' and 'Code Versions' buttons, and a search bar.

The code editor displays the following SQL query:

```
64
65
66
67
68
69
70
71
72
73
74
```

--6. Calculate the total number of purchases per location, filtering for payment methods that are not null.  
group by location  
having count(purchase\_amount) > 5;

--7. Create a column Spender Category that classifies customers using CASE:  
--'High' if amount > 80. 'Medium' if BETWEEN 50 AND 80.

The results table shows the total number of purchases for each location:

LOCATION	TOTAL_PURCHASES
Maine	34
Kentucky	27
null	21
New York	27
Oregon	24
Rhode Island	26
Florida	29
Massachusetts	20

## Q7

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

```
72  --7. Create a column Spender Category that classifies customers using CASE:  
73  --'High' if amount > 80, 'Medium' if BETWEEN 50 AND 80.  
74  --'Low' otherwise. Replace NULLs in purchase_amount with 0.  
75  --Expected Columns: Customer ID, purchase_amount, Spender Category  
76  select customer_id,  
77      coalesce(purchase_amount,0) as purchase_amount,  
78      case  
79          when coalesce(purchase_amount,0) > 80 then 'High Spender'  
80          when coalesce(purchase_amount,0) between 50 and 80 then 'Medium Spender'  
81          else 'Low Spender'  
82      end as SPENDER_CATEGORY  
83  from shopping_tread;
```

The results table has three columns: CUSTOMER\_ID, PURCHASE\_AMOUNT, and SPENDER\_CATEGORY. The data is as follows:

CUSTOMER_ID	PURCHASE_AMOUNT	SPENDER_CATEGORY
120	120	99.0 High Spender
121	121	60.0 Medium Spender
122	122	37.0 Low Spender
123	123	0.0 Low Spender
124	124	0.0 Low Spender
125	125	28.0 Low Spender

Query Details: Query duration 57ms, Rows 300, Query ID 0fc02bc8-000c-b2eb-0...

## Q8

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

```
87  --8. Find customers who have no Previous Purchases value but whose Color is not NULL.  
88  --Expected Columns: Customer ID, Color, Previous Purchases  
89  select customer_id,  
90      color,  
91      previous_purchases  
92  from shopping_tread  
93  where previous_purchases is null  
94  and color is not null;  
95
```

The results table has three columns: CUSTOMER\_ID, COLOR, and PREVIOUS\_PURCHASES. The data is as follows:

CUSTOMER_ID	COLOR	PREVIOUS_PURCHASES
226	Black	null
230	Green	null
251	Gray	null
253	Green	null
260	Pink	null
271	Yellow	null
282	Pink	null
283	Green	null
290	White	null

## Q9

The screenshot shows a database query editor interface with multiple tabs at the top: Practical 1, Practical 2, Practical 3, Bright\_Coffee\_Case\_study 1, BrightTV\_Case\_study 2, and a new tab. The current tab is Bright\_Coffee\_Case\_study 1. The code in the editor is:

```
96 --9. Group records by Frequency of Purchases and show the total amount spent per group, treating NULL frequencies as 'Unknown'.
97 --Expected Columns: Frequency of Purchases, Total purchase_amount
98 select
99     coalesce(frequency_of_purchases, 'Unknown') as frequency_of_purchases,
100    sum(purchase_amount) as total_amount_spent_per_group
101
102 from shopping_tread
103 group by frequency_of_purchases;
```

The results table has two columns: FREQUENCY\_OF\_PURCHASES and TOTAL\_AMOUNT\_SPENT\_PER\_GROUP. The data is:

FREQUENCY_OF_PURCHASES	TOTAL_AMOUNT_SPENT_PER_GROUP
Every 3 Months	1749.0
Weekly	2184.0
Bi-Weekly	2099.0
Monthly	1780.0
Unknown	1518.0
Fortnightly	2033.0
Annually	1765.0
Quarterly	2541.0

## Q10

The screenshot shows a database query editor interface with multiple tabs at the top: Practical 1, Practical 2, Practical 3, Bright\_Coffee\_Case\_study 1, BrightTV\_Case\_study 2, and a new tab. The current tab is BrightTV\_Case\_study 2. The code in the editor is:

```
112
113 --11. Return the top 5 Locations with the highest total purchase_amount, replacing NULLs in amount with 0.
114 --Expected Columns: Location, Total purchase_amount
115 select location,
116     sum(coalesce(purchase_amount, 0)) as total_purchase_amount
117 from shopping_tread
118 group by location
119 order by total_purchase_amount desc
120 limit 5;
121
```

The results table has two columns: LOCATION and TOTAL\_PURCHASE\_AMOUNT. The data is:

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

## Q11

PRACTICALS.PRACTICAL\_3

```
108     count(item_purchased) as total_purchases
109   from shopping_tread
110  where category is not null
111  group by category;
112
113 --11. Return the top 5 Locations with the highest total purchase_amount, replacing NULLs in amount with 0.
114 --Expected Columns: Location, Total purchase_amount
115   select location,
116         sum(coalesce(purchase_amount, 0)) as total_purchase_amount
117   from shopping_tread
118  group by location
119  order by total_purchase_amount desc
120  limit 5;
```

Results

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

Query Details

- Query duration: 29ms
- Rows: 5
- Query ID: 01c031c1-000c-b2ea-0...

## Q12

PRACTICALS.PRACTICAL\_3

```
122 --12. Group customers by Gender and Size, and count how many entries have a NULL Color.
123 --Expected Columns: Gender, Size, Null Color Count
124
125   select gender,
126         size,
127         count_if(color is null) as Null_Color_Count
128   from shopping_tread
129  group by gender,
130        size;
```

Results

GENDER	SIZE	NULL_COLOR_COUNT
Male	null	6
Male	M	7
Male	L	6
Male	XL	5
Male	S	5

Query Details

- Query duration: 25ms
- Rows: 5
- Query ID: 01c031c2-000c-b2ea-0...

## Q13

PRACTICALS.PRACTICAL\_3

```
132  --13. Identify all Item Purchased where more than 3 purchases had NULL Shipping Type.
133  --Expected Columns: Item Purchased, NULL Shipping Type Count
134
135  > select item_purchased,
136      count_if(shipping_type is null) as NULL_Shipping_Type_Count
137  from shopping_tread
138  group by item_purchased
139  having count_if(shipping_type is null) > 3;
140
```

Results

ITEM_PURCHASED	NULL_Shipping_Type_Count
null	4
Shirt	5
Shoes	4

## Q14

PRACTICALS.PRACTICAL\_3

```
135  > select item_purchased,
136      count_if(shipping_type is null) as NULL_Shipping_Type_Count
137  from shopping_tread
138  group by item_purchased
139  having count_if(shipping_type is null) > 3;
140
141  --14. Show a count of how many customers per Payment Method have NULL Review Rating.
142  --Expected Columns: Payment Method, Missing Review Rating Count
143
144  > select payment_method,
145      count_if(review_rating is null) as Missing_Review_rating_count
146  from shopping_tread
147  group by payment_method;
148
149
```

Results

PAYMENT_METHOD	MISSING REVIEW RATING COUNT
Credit Card	8
PayPal	3
Debit Card	7
null	2
Cash	4

## Q15

The screenshot shows a database query editor interface. The top navigation bar includes tabs for Practical 1, Practical 2, Practical 3, Bright\_Coffee\_Case\_study 1, BrightTV\_Case\_study 2, and a plus sign for creating new workspaces. On the right, there are buttons for ACCOUNTADMIN, COMPUTE\_WH (X-Small), Share, and a play icon. The main area displays a code editor with the following SQL query:

```
149 -----
150 --15. Group by Category and return the average Review Rating, replacing NULLs with 0, and filter only where average is greater than 3.5.
151 --Expected Columns: Category, Average Review Rating
152     select category,
153         avg(coalesce(review_rating, 0)) as Average_Review_Rating
154     from shopping_tread
155     group by category
156     having avg(coalesce(review_rating, 0)) > 3.5;
```

Below the code editor, there are two tabs: 'Results' (selected) and 'Chart'. The results table has columns 'CATEGORY' and 'AVERAGE REVIEW RATING'. A message at the top of the results table says 'Query produced no results'.

## Q16

The screenshot shows a database query editor interface. The top navigation bar includes tabs for Practical 1, Practical 2, Practical 3, Bright\_Coffee\_Case\_study 1, BrightTV\_Case\_study 2, and a plus sign for creating new workspaces. On the right, there are buttons for ACCOUNTADMIN, COMPUTE\_WH (X-Small), Share, and a play icon. The main area displays a code editor with the following SQL query:

```
154 -----
155     select category,
156         avg(coalesce(review_rating, 0)) > 3.5;
157 -----
158 --16. List all Colors that are missing (NULL) in at least 2 rows and the average Age of customers for those rows.
159 --Expected Columns: Color, Average Age
160     select color,
161         avg(age) as Average_Age
162     from shopping_tread
163     group by color
164     having count_if(color is null) >= 2;
```

Below the code editor, there are two tabs: 'Results' (selected) and 'Chart'. The results table has columns 'COLOR' and 'AVERAGE AGE'. The table contains one row with 'null' in the COLOR column and '47.8461538' in the AVERAGE AGE column.

## Q17

The screenshot shows a database query editor interface with several tabs at the top: Practical 1, Practical 2, Practical 3, Bright\_Coffee\_Case\_study 1, BrightTV\_Case\_study 2, and a new tab button. The current tab is Bright\_Coffee\_Case\_study 1. The user is in the PRACTICALS.PRACTICAL\_3 workspace. The code editor contains the following SQL query:

```
164 having count_if(color is null) >= 2;
165
166 --17. Use CASE to create a column Delivery Speed: 'Fast' if Shipping Type is 'Express' or
167 --'Next Day Air', 'Slow' if 'Standard', 'Other' for all else including NULL. Then count how many customers fall into each category.
168 --Expected Columns: Delivery Speed, Customer Count
169
170 select
171   case
172     when shipping_type in ('Express','Next Day Air') then 'Fast'
173     when shipping_type = 'Standard' then 'Slow'
174     else 'Other'
175   end as Delivery_Speed,
176
177   count(*) as customer_count
178
179 from shopping_tread
180 group by Delivery_Speed;
```

The results table has two columns: DELIVERY\_SPEED and CUSTOMER\_COUNT. The data is as follows:

DELIVERY_SPEED	CUSTOMER_COUNT
Other	166
Fast	89
Slow	45

## Q18

The screenshot shows a database query editor interface with several tabs at the top: Practical 1, Practical 2, Practical 3, Bright\_Coffee\_Case\_study 1, BrightTV\_Case\_study 2, and a new tab button. The current tab is Bright\_Coffee\_Case\_study 1. The user is in the PRACTICALS.PRACTICAL\_3 workspace. The code editor contains the following SQL query:

```
177
178 from shopping_tread
179 group by Delivery_Speed;
180
181
182 --18. Find customers whose purchase_amount is NULL and whose Promo Code Used is 'Yes'.
183 --Expected Columns: Customer ID, purchase_amount, Promo Code Used
184
185 select customer_id,
186   purchase_amount,
187   promo_code_used
188 from shopping_tread
189 where purchase_amount is null
190 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
1	13	null TRUE
2	30	null TRUE
3	78	null TRUE
4	95	null TRUE
5	124	null TRUE
6	129	null TRUE

## Q19

PRACTICALS.PRACTICAL\_3

```
186     promo_code_used
187     from shopping_tread
188     where purchase_amount is null
189     and promo_code_used = 'Yes';
190
--19. Group by Location and show the maximum Previous Purchases, replacing NULLs with 0, only where the average rating is above 4.0.
191 --Expected Columns: Location, Max Previous Purchases, Average Review Rating
192
193     select location,
194         max(coalesce(previous_purchases, 0)) as Max_Previous_Purchases,
195         avg(review_rating) as Average_Review_Rating
196     from shopping_tread
197     group by location;
```

Results

LOCATION	MAX_PREVIOUS_PURCHASES	AVERAGE REVIEW RATING
Rhode Island	50.0	3.7428571
Kentucky	46.0	3.7107143
Texas	47.0	3.5523810
Massachusetts	47.0	3.6580645
null	50.0	3.5956522
Oregon	50.0	3.6133333
New York	49.0	3.9280000

## Q20

PRACTICALS.PRACTICAL\_3

```
195     avg(review_rating) as Average_Review_Rating
196     from shopping_tread
197     group by location;
198
--20. Show customers who have a NULL Shipping Type but made a purchase in the range of 30 to 70 USD.
199 --Expected Columns: Customer ID, Shipping Type, purchase_amount, Item Purchased
200
201     select customer_id,
202         shipping_type,
203         purchase_amount,
204         item_purchased
205
206     from shopping_tread
207     where shipping_type is null
208     and purchase_amount between 30 and 70;
```

Results

CUSTOMER_ID	SHIPPING_TYPE	PURCHASE_AMOUNT	ITEM_PURCHASED
1	15	54.0	Jeans
2	105	43.0	Shirt
3	141	37.0	Shorts
4	196	66.0	Coat
5	213	36.0	Shirt