

Sindiswa Jachin Shongwe

Practical 1: BrightLight Data Analytics Coding Practical Basic SQL Syntax

Question 1

-- Q1. Display all columns for all transactions.

-- Expected output: All columns

The screenshot shows a BrightLight Data Analytics workspace interface. At the top, there are two tabs: '2025-10-15 9:38pm' and '2025-10-15 9:42pm'. Below the tabs is a toolbar with a plus sign, a search icon, and user information: 'ACCOUNTADMIN COMPUTE_WH (X-Small) Share'. The main area has a dark background with white text. It displays a SQL query and its results. The query is:

```
1 --Q1. Display all columns for all transactions.  
2 -- Expected output: All columns  
3  
4 SELECT *  
5 FROM practical1.retail.retail_sale_dataset;
```

The results section shows a table with 9 rows of data. The columns are: TRANSACTION_ID, DATE, CUSTOMER_ID, GENDER, AGE, PRODUCT_CATEGORY, QUANTITY, PRICE_PER_UNIT, and TOTAL_AMOUNT. The data is as follows:

	TRANSACTION_ID	DATE	CUSTOMER_ID	GENDER	AGE	PRODUCT_CATEGORY	QUANTITY	PRICE_PER_UNIT	TOTAL_AMOUNT
1	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
2	2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
3	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
4	4	2023-05-21	CUST004	Male	37	Clothing	1	500	500
5	5	2023-05-06	CUST005	Male	30	Beauty	2	50	100
6	6	2023-04-25	CUST006	Female	45	Beauty	1	30	30
7	7	2023-03-13	CUST007	Male	46	Clothing	2	25	50
8	8	2023-02-22	CUST008	Male	30	Electronics	4	25	100
9	9	2023-12-13	CUST009	Male	63	Electronics	2	300	600

Question 2

- Q2. Display only the Transaction ID, Date, and Customer ID for all records.
- Expected output: Transaction ID, Date, Customer ID

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

Query Editor:

```
1  -- 01. Display all columns for all transactions.  
2  -- Expected output: All columns  
3  SELECT *  
4  FROM practical1.retail.retail_sale_dataset;  
5  
6  -- 02. Display only the Transaction ID, Date, and Customer ID for all records.  
7  -- Expected output: Transaction ID, Date, Customer ID  
8  
9  SELECT TRANSACTION_ID,  
10   DATE,  
11   CUSTOMER_ID  
12  FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
```

Results Table:

#	TRANSACTION_ID	DATE	CUSTOMER_ID
1		2023-11-24	CUST001
2		2023-02-27	CUST002
3		2023-01-13	CUST003
4		2023-05-21	CUST004
5		2023-05-06	CUST005
6		2023-04-25	CUST006

Query Details:

- Query duration: 408ms
- Rows: 1K
- Query ID: 01bfbbaa-000c-b0ed-0...

Question 3

-- Q3. Display all the distinct product categories in the dataset.

-- Expected output: Product Category

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

- Timestamps: 2025-10-15 9:38pm and 2025-10-15 9:42pm.
- User: ACCOUNTADMIN - COMPUTE_WH (X-Small).
- Query Editor:
 - No Database selected.
 - Settings dropdown.
 - Code area:

```
7
8 -- Q2. Display only the Transaction ID, Date, and Customer ID for all records.
9 -- Expected output: Transaction ID, Date, Customer ID
10
11 SELECT TRANSACTION_ID,
12     DATE,
13     CUSTOMER_ID
14 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
15
16
17 -- Q3. Display all the distinct product categories in the dataset.
18 -- Expected output: Product Category
19
20 SELECT DISTINCT product_category
21 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
```
 - Results tab is selected.
- Results Table:

PRODUCT_CATEGORY
1 Clothing
2 Beauty
3 Electronics
- Query Details:
 - Query duration: 66ms.
 - Rows: 3.
 - Query ID: 01bfb00-000c-b0ed-0...
- Show more button.

-- Q4. Display all the distinct gender values in the dataset.

-- Expected output: Gender

2025-10-15 9:38pm 2025-10-15 9:42pm + -

No Database selected Settings ↗ Open in Workspaces 🔍

```
12     DATE,  
13     CUSTOMER_ID  
14 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;  
15  
16 -- Q3. Display all the distinct product categories in the dataset.  
17 -- Expected output: Product Category  
18  
19 SELECT DISTINCT product_category  
20 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;  
21  
22 -- Q4. Display all the distinct gender values in the dataset.  
23 -- Expected output: Gender  
24  
25 SELECT DISTINCT GENDER  
26 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
```

↳ Results ↵ Chart 🔍 ⚡ ⏪ ⏴ ⏵ ⏹ ⏷

	GENDER
1	Male
2	Female

Query Details ...
Query duration 69ms
Rows 2
Query ID 01bfb8-000c-b0ed-0...
Show more ↴

-- Q5. Display all transactions where the Age is greater than 40.

-- Expected output: All columns

2025-10-15 9:38pm | 2025-10-15 9:42pm | + | - |

No Database selected | Settings |

ACCOUNTADMIN COMPUTE_WH (X-Small) Share

```
19
20 SELECT DISTINCT product_category
21 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
22
23 -----
24 -- Q4. Display all the distinct gender values in the dataset.
25 -- Expected output: Gender
26 SELECT DISTINCT GENDER
27 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
28 -----
29 -- Q5. Display all transactions where the Age is greater than 40.
30 -- Expected output: All columns
31
32 SELECT *
33 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET
34 WHERE AGE > 40;
35 -----
```

-- Q6. Display all transactions where the Price per Unit is between 100 and 500.

-- Expected output: All columns

The screenshot shows a database interface with two tabs at the top: '2025-10-15 9:38pm' and '2025-10-15 9:42pm'. The code editor contains the following SQL queries:

```
27 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
28
29 -- Q5. Display all transactions where the Age is greater than 40.
30 -- Expected output: All columns
31
32 SELECT *
33 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET
34 WHERE AGE > 40;
35
36 -- Q6. Display all transactions where the Price per Unit is between 100 and 500.
37 -- Expected output: All columns
38
39 SELECT *
40 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET
41 WHERE PRICE_PER_UNIT BETWEEN 100 AND 500;
```

The results tab is selected, displaying the following data:

#	TRANSACTION_ID	DATE	CUSTOMER_ID	GENDER	AGE	PRODUCT_CATEGORY	QUANTITY	PRICE_PER_UNIT	TOTAL_AMOUNT
1	2	2023-02-27	CUST002	Female	26	Clothing	2	500	1000
2	4	2023-05-21	CUST004	Male	37	Clothing	1	500	500
3	9	2023-12-13	CUST009	Male	63	Electronics	2	300	600
4	13	2023-08-05	CUST013	Male	22	Electronics	3	500	1500
5	15	2023-01-16	CUST015	Female	42	Electronics	4	500	2000

-- Q7. Display all transactions where the Product Category is either 'Beauty' or 'Electronics'.

--Expected output: All columns

The screenshot shows a database interface with two tabs: 'Results' and 'Chart'. The 'Results' tab is selected, displaying a table with five rows of transaction data. The table has the following columns: TRANSACTION_ID, DATE, CUSTOMER_ID, GENDER, AGE, PRODUCT_CATEGORY, QUANTITY, PRICE_PER_UNIT, and TOTAL_AMOUNT. The data is as follows:

	# TRANSACTION_ID	⌚ DATE	▲ CUSTOMER_ID	▲ GENDER	# AGE	▲ PRODUCT_CATEGORY	# QUANTITY	# PRICE_PER_UNIT	# TOTAL_AMOUNT
1	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
2	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
3	5	2023-05-06	CUST005	Male	30	Beauty	2	50	100
4	6	2023-04-25	CUST006	Female	45	Beauty	1	30	30
5	8	2023-02-22	CUST008	Male	30	Electronics	4	25	100

The code editor above the results shows two queries:

```
55  
56  
57 -- Q6. Display all transactions where the Price per Unit is between 100 and 500.  
58 -- Expected output: All columns  
59  
60 SELECT *  
61 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET  
62 WHERE PRICE_PER_UNIT BETWEEN 100 AND 500;  
63  
64 -- Q7. Display all transactions where the Product Category is either 'Beauty' or 'Electronics'.  
65 --Expected output: All columns  
66  
67 SELECT *  
68 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET  
69 WHERE PRODUCT_CATEGORY='Beauty' OR PRODUCT_CATEGORY='Electronics';
```

-- Q8. Display all transactions where the Product Category is not 'Clothing'.

-- Expected output: All columns

The screenshot shows a database query editor interface. At the top, there are two tabs: '2025-10-15 9:38pm' and '2025-10-15 9:42pm'. Below the tabs, the main area has a dark background with white text. The code is as follows:

```
42 WHERE PRICE_PER_UNIT BETWEEN 100 AND 500;
43
44 -----
45 -- Q7. Display all transactions where the Product Category is either 'Beauty' or 'Electronics'.
46 --Expected output: All columns
47
48 SELECT *
49 FROM PRACTICAL1.RETAIL.RETAIL_SALE_DATASET
50 WHERE PRODUCT_CATEGORY='Beauty' OR PRODUCT_CATEGORY='Electronics';
51
52 -----
53 -- Q8. Display all transactions where the Product Category is not 'Clothing'.
54 -- Expected output: All columns
55
56 SELECT *
57 FROM PRACTICAL1.RETAIL.RETAIL_SALE_DATASET
58 WHERE PRODUCT_CATEGORY !='Clothing';
```

Below the code, there are two tabs: 'Results' (which is selected) and 'Chart'. The 'Results' tab displays a table with the following data:

	# TRANSACTION_ID	⌚ DATE	▲ CUSTOMER_ID	▲ GENDER	# AGE	▲ PRODUCT_CATEGORY	# QUANTITY	# PRICE_PER_UNIT	# TOTAL_AMOUNT
1	1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
2	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
3	5	2023-05-06	CUST005	Male	30	Beauty	2	50	100
4	6	2023-04-25	CUST006	Female	45	Beauty	1	30	30
5	8	2023-02-22	CUST008	Male	30	Electronics	4	25	100

-- Q9. Display all transactions where the Quantity is greater than or equal to 3.

-- Expected output: All columns

The screenshot shows a Databricks notebook interface. The top navigation bar includes '2025-10-15 9:38pm' and '2025-10-15 9:42pm'. The workspace dropdown shows 'PRATICAL1.RETAIL'. The top right features 'ACCOUNTADMIN', 'COMPUTE_WH (X-Small)', 'Share', and a play button. Below the header is a search bar and a 'Open in Workspaces' link. The code editor contains the following SQL queries:

```
50 WHERE PRODUCT_CATEGORY = 'Beauty' OR PRODUCT_CATEGORY = 'Electronics';
51
52 -- Q8. Display all transactions where the Product Category is not 'Clothing'.
53 -- Expected output: All columns
54
55 SELECT *
56 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET
57 WHERE PRODUCT_CATEGORY != 'Clothing';
58
59 -- Q9. Display all transactions where the Quantity is greater than or equal to 3.
60 -- Expected output: All columns
61
62 SELECT *
63 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET
64 WHERE QUANTITY >= 3;
```

The results section shows a table with the following data:

# TRANSACTION_ID	DATE	CUSTOMER_ID	GENDER	AGE	PRODUCT_CATEGORY	QUANTITY	PRICE_PER_UNIT	TOTAL_AMOUNT
1	2023-11-24	CUST001	Male	34	Beauty	3	50	150
2	2023-02-22	CUST008	Male	30	Electronics	4	25	100
3	2023-10-07	CUST010	Female	52	Clothing	4	50	200
4	2023-10-30	CUST012	Male	35	Beauty	3	25	75
5	2023-08-05	CUST013	Male	22	Electronics	3	500	1500

-- Q10. Count the total number of transactions.

--Expected output: Total_Transactions

The screenshot shows a Databricks notebook interface. The top navigation bar includes '2025-10-15 9:38pm' and 'Practical 1'. The workspace dropdown shows 'PRATICAL1.RETAIL'. The top right features 'ACCOUNTADMIN', 'COMPUTE_WH (X-Small)', 'Share', and a play button. Below the header is a search bar and a 'Open in Workspaces' link. The code editor contains the following SQL queries:

```
5 / FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET
58 WHERE PRODUCT_CATEGORY != 'Clothing';
59
60 -- Q9. Display all transactions where the Quantity is greater than or equal to 3.
61 -- Expected output: All columns
62
63 SELECT *
64 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET
65 WHERE QUANTITY >= 3;
66
67 -- Q10. Count the total number of transactions.
68 -- Expected output: Total_Transactions
69 SELECT COUNT(TRANSACTION_ID) AS Total_Transaction
70 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
```

The results section shows a table with the following data:

TOTAL_TRANSACTION
1 1000

-- Q11. Find the average Age of customers.

-- Expected output: Average_Age

The screenshot shows a SQL editor interface with the following details:

- Timestamp: 2025-10-15 9:38pm
- Workspace: Practical 1
- Dataset: PRATICAL1.RETAIL
- User: ACCOUNTADMIN
- Compute: COMPUTE_WH (X-Small)
- Share button
- Open in Workspaces
- Search icon

The code in the editor is:

```
63  
64 SELECT *  
65 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET  
66 WHERE QUANTITY >= 3;  
67  
68 -----  
69 -- Q10. Count the total number of transactions.  
70 --Expected output: Total_Transactions  
71 SELECT COUNT(TRANSACTION_ID) AS Total_Transaction  
FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;  
72  
73 -----  
74 -- Q11. Find the average Age of customers.  
75 -- Expected output: Average_Age  
76  
77 SELECT AVG(AGE) AS Average_Age  
FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
```

The results pane shows:

AVERAGE_AGE	
1	41.392000

-- Q12. Find the total quantity of products sold.

-- Expected output: Total_Quantity

The screenshot shows a SQL editor interface with the following details:

- Timestamp: 2025-10-15 9:38pm
- Workspace: Practical 1
- Dataset: PRATICAL1.RETAIL
- User: ACCOUNTADMIN
- Compute: COMPUTE_WH (X-Small)
- Share button
- Open in Workspaces
- Search icon

The code in the editor is:

```
69 -----  
70 -- Q10. Count the total number of transactions.  
71 --Expected output: Total_Transactions  
72 SELECT COUNT(TRANSACTION_ID) AS Total_Transaction  
FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;  
73  
74 -----  
75 -- Q11. Find the average Age of customers.  
76 -- Expected output: Average_Age  
77  
78 SELECT AVG(AGE) AS Average_Age  
FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;  
79  
80 -----  
81 -- Q12. Find the total quantity of products sold.  
82 -- Expected output: Total_Quantity  
83 SELECT COUNT(QUANTITY) AS Total_Quantity  
FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
```

The results pane shows:

TOTAL_QUANTITY	
1	1000

-- Q13. Find the maximum Total Amount spent in a single transaction.

-- Expected output: Max_Total_Amount

The screenshot shows a database query editor interface. The top bar displays the date and time (2025-10-15 9:38pm), the workspace name (Practical 1), and the account information (ACCOUNTADMIN COMPUTE_WH (X-Small)). The main area shows a code editor with SQL queries numbered 75 to 91. The last query is:

```
90 | SELECT MAX(TOTAL_AMOUNT) AS Max_Total_Amount
91 | FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
```

The results pane below shows the output for the last query:

MAX_TOTAL_AMOUNT
2000

-- Q14. Find the minimum Price per Unit in the dataset.

-- Expected output: Min_Price_per_Unit

The screenshot shows a database query editor interface. The top bar displays the date and time (2025-10-15 9:38pm), the workspace name (Practical 1), and the account information (ACCOUNTADMIN COMPUTE_WH (X-Small)). The main area shows a code editor with SQL queries numbered 81 to 96. The last query is:

```
93 | SELECT MIN(PRICE_PER_UNIT) AS MIN_PRICE_PER_UNIT
94 | FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
95 |
96 | -- Q14. Find the minimum Price per Unit in the dataset.
97 | -- Expected output: Min_Price_per_Unit
```

The results pane below shows the output for the last query:

MIN_PRICE_PER_UNIT
25

-- Q15. Find the number of transactions per Product Category.

-- Expected output: Product Category, Transaction_Count

The screenshot shows a Jupyter Notebook cell with the following content:

```
2025-10-15 9:38pm Practical1 + - ACCOUNTADMIN COMPUTE_WH (X-Small) Share Open in Workspaces
```

PRATICAL1.RETAIL Settings

```
8/ -- Q15. Find the maximum total Amount spent in a single transaction.
88 -- Expected output: Max.Total_Amount
89
90 SELECT MAX(TOTAL_AMOUNT) AS Max_Total_Amount
91 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
92
93 -- Q14. Find the minimum Price per Unit in the dataset.
94 -- Expected output: Min.Price_per_Unit
95 SELECT MIN(PRICE_PER_UNIT) AS MIN_PRICE_PER_UNIT
96 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
97
98 -- Q15. Find the number of transactions per Product Category.
99 -- Expected output: Product Category, Transaction_Count
100 SELECT PRODUCT_CATEGORY, COUNT(TRANSACTION_ID) AS Transaction_Count
101 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET
102 GROUP BY PRODUCT_CATEGORY;
```

Results

PRODUCT_CATEGORY	TRANSACTION_COUNT
Beauty	307
Clothing	351
Electronics	342

-- Q16. Find the total revenue (Total Amount) per gender.

-- Expected output: Gender, Total_Revenue

The screenshot shows a database query editor interface. The top bar displays the date and time (2025-10-15 9:38pm), workspace name (Practical 1), and account information (ACCOUNTADMIN, COMPUTE_WH (X-Small)). The main area shows a code editor with SQL queries numbered 98 to 114. Queries 103 and 111 are highlighted. The results pane below shows a table with two rows: Male and Female, each with a value for Total_Revenue.

A GENDER	# TOTAL_REVENUE
1 Male	223160
2 Female	232840

-- Q17. Find the average Price per Unit per product category.

-- Expected output: Product Category, Average_Price

The screenshot shows a database query editor interface. The top bar displays the date and time (2025-10-15 9:38pm), workspace name (Practical 1), and account information (ACCOUNTADMIN, COMPUTE_WH (X-Small)). The main area shows a code editor with SQL queries numbered 104 to 120. Queries 111 and 118 are highlighted. The results pane below shows a table with three rows: Beauty, Clothing, and Electronics, each with a value for Average_Price_Per_Unit.

A PRODUCT_CATEGORY	# AVERAGE_PRICE_PER_UNIT
1 Beauty	184.055375
2 Clothing	174.287749
3 Electronics	181.900585

-- Q18. Find the total revenue per product category where total revenue is greater than 10,000.

-- Expected output: Product Category, Total_Revenue

The screenshot shows a Jupyter Notebook cell with the following content:

```
2025-10-15 9:38pm Practical 1 + v ACCOUNTADMIN COMPUTE_WH (X-Small) Share Open in Workspaces PRATICAL1.RETAIL Settings ▾
111 SELECT GENDER,SUM(TOTAL_AMOUNT) AS Total_Revenue
112 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET
113 GROUP BY GENDER;
114 -----
115 -- Q17. Find the average Price per Unit per product category.
116 -- Expected output: Product Category, Average_Price
117
118 SELECT PRODUCT_CATEGORY, AVG(PRICE_PER_UNIT) AS Average_Price_Per_Unit
119 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET
120 GROUP BY PRODUCT_CATEGORY;
121 -----
122 -- Q18. Find the total revenue per product category where total revenue is greater than 10,000.
123 -- Expected output: Product Category, Total_Revenue
124 SELECT PRODUCT_CATEGORY,SUM(TOTAL_AMOUNT) AS Total_Revenue
125 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET
126 GROUP BY PRODUCT_CATEGORY
127 HAVING Total_Revenue > 10000;
```

The Results tab is selected, displaying the following table:

PRODUCT_CATEGORY	TOTAL_REVENUE
Beauty	143515
Clothing	155580
Electronics	156905

-- Q19. Find the average quantity per product category where the average is more than 2.

-- Expected output: Product Category, Average_Quantity

The screenshot shows a Jupyter Notebook cell with the following content:

```
2025-10-15 9:38pm Practical 1 + ▾
```

ACCOUNTADMIN COMPUTE_WH (X-Small) Share Open in Workspaces

```
PRATICAL1.RETAIL Settings ▾
```

```
121 -- Q18. Find the total revenue per product category where total revenue is greater than 10,000.
122 -- Expected output: Product Category, Total_Revenue
123
124
125 SELECT PRODUCT_CATEGORY, SUM(TOTAL_AMOUNT) AS Total_Revenue
126 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET
127 GROUP BY PRODUCT_CATEGORY
128 HAVING Total_Revenue > 10000;
129
130 -- Q19. Find the average quantity per product category where the average is more than 2.
131 -- Expected output: Product Category, Average_Quantity
132
133 SELECT PRODUCT_CATEGORY, AVG(QUANTITY) AS Average_Quantity
134 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET
135 GROUP BY PRODUCT_CATEGORY
136 HAVING Average_Quantity > 2;
```

Results

PRODUCT_CATEGORY	AVERAGE_QUANTITY
Beauty	2.511401
Clothing	2.547009
Electronics	2.482456

-- Q20. Display a column called Spending_Level that shows 'High' if Total Amount > 1000, otherwise 'Low'.

-- Expected output: Transaction ID, Total Amount, Spending_Level

The screenshot shows a Jupyter Notebook cell with the following content:

```
2025-10-15 9:38pm Practical 1 + ▾
```

ACCOUNTADMIN COMPUTE_WH (X-Small) Share Open in Workspaces

```
PRATICAL1.RETAIL Settings ▾
```

```
152
153 SELECT PRODUCT_CATEGORY, AVG(QUANTITY) AS Average_Quantity
154 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET
155 GROUP BY PRODUCT_CATEGORY
156 HAVING Average_Quantity > 2;
157
158 -- Q20. Display a column called Spending_Level that shows 'High' if Total Amount > 1000, otherwise 'Low'.
159 -- Expected output: Transaction ID, Total Amount, Spending_Level
160
161 SELECT Transaction_ID,
162       TOTAL_AMOUNT,
163       CASE
164         WHEN TOTAL_AMOUNT > 1000 THEN 'High'
165         ELSE 'Low'
166       END AS Spending_Level
167 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
```

Results

TRANSACTION_ID	TOTAL_AMOUNT	SPENDING_LEVEL
1	150	Low
2	1000	Low
3	30	Low
4	500	Low
5	100	Low

-- Q21. Display a new column called Age_Group that labels customers as:

- • 'Youth' if Age < 30
- • 'Adult' if Age is between 30 and 59
- • 'Senior' if Age >= 60

-- Expected output: Customer ID, Age, Age_Group

The screenshot shows a data processing interface with a query editor at the top and a results table below. The query editor contains the following code:

```
2025-10-15 9:38pm Practical 1 + v ACCOUNTADMIN COMPUTE_WH (X-Small) Share Open in Workspaces PRATICAL1.RETAIL Settings ▾
146 END AS Spending_Level
147 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
148
149 -- Q21. Display a new column called Age_Group that labels customers as:
150 -- . 'Youth' if Age < 30
151 -- . 'Adult' if Age is between 30 and 59
152 -- . 'Senior' if Age >= 60
153 -- Expected output: Customer ID, Age, Age_Group
154
155 SELECT CUSTOMER_ID,AGE,
156 CASE
157 WHEN AGE < 30 THEN 'Youth'
158 WHEN AGE BETWEEN 30 AND 59 THEN 'Adult'
159 WHEN AGE >=60 THEN 'Senior'
160 END AS AGE_GROUP
161 FROM PRATICAL1.RETAIL.RETAIL_SALE_DATASET;
162
```

The results table below the editor shows the following data:

	CUSTOMER_ID	AGE	AGE_GROUP
1	CUST001	34	Adult
2	CUST002	26	Youth
3	CUST003	50	Adult
4	CUST004	37	Adult
5	CUST005	30	Adult