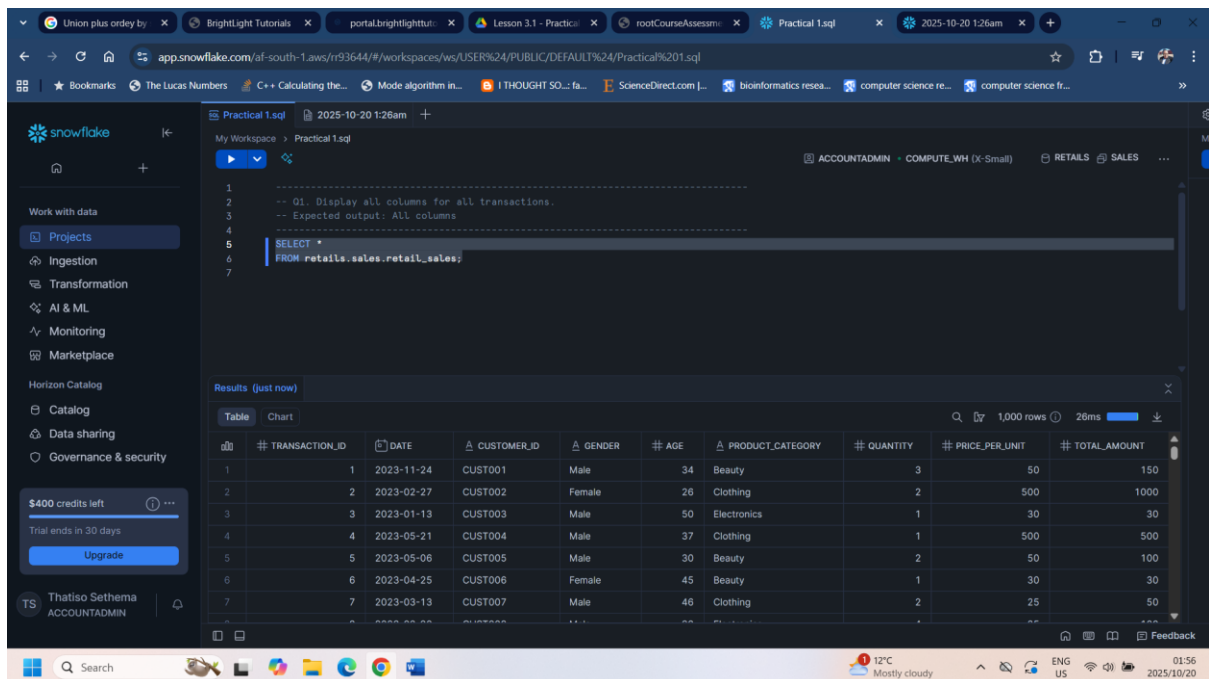


Thatiso Sethema

Practical 1: Basic SQL Syntax

Question 1



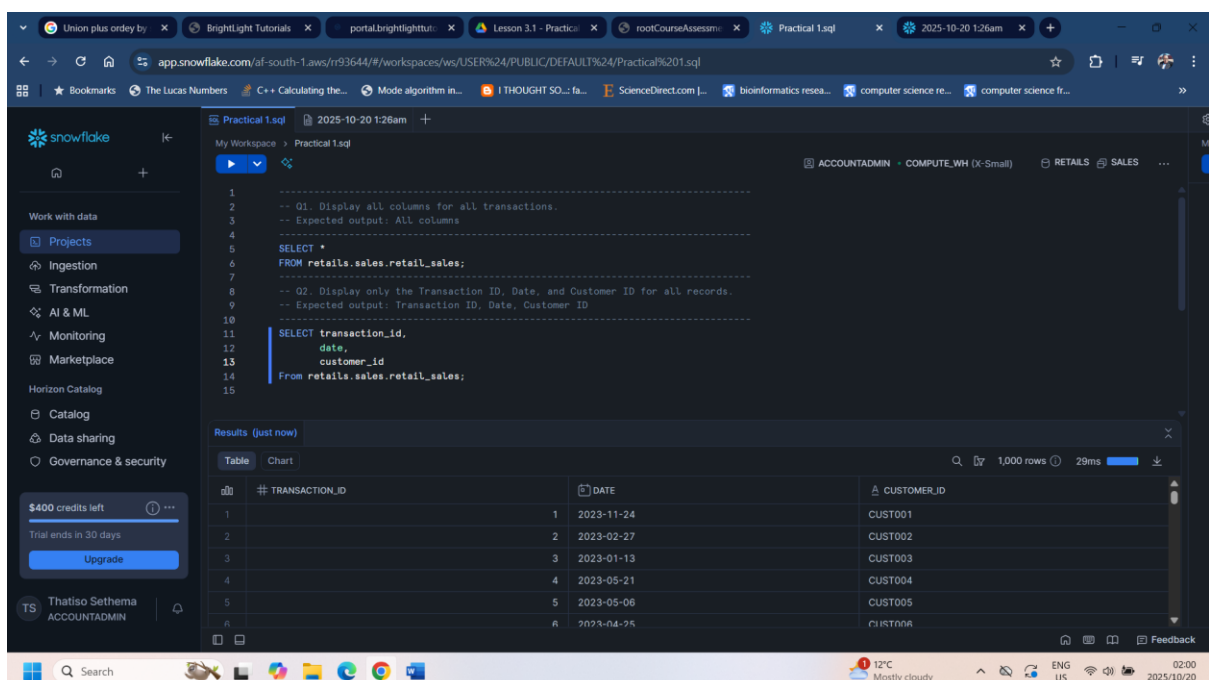
The screenshot shows the Snowflake SQL editor interface. The query editor contains the following SQL code:

```
1 -----  
2 -- Q1. Display all columns for all transactions.  
3 -- Expected output: All columns  
4 -----  
5 SELECT *  
6 FROM retail_sales.retail_sales;  
7
```

The results pane shows the following data:

Table	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

Question 2



The screenshot shows the Snowflake SQL editor interface. The query editor contains the following SQL code:

```
1 -----  
2 -- Q1. Display all columns for all transactions.  
3 -- Expected output: All columns  
4 -----  
5 SELECT *  
6 FROM retail_sales.retail_sales;  
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 retail_sales.retail_sales;  
15
```

The results pane shows the following data:

Table	TRANSACTION_ID	DATE	CUSTOMER_ID
1	1	2023-11-24	CUST001
2	2	2023-02-27	CUST002
3	3	2023-01-13	CUST003
4	4	2023-05-21	CUST004
5	5	2023-05-06	CUST005
6	6	2023-04-25	CUST006
7	7	2023-03-13	CUST007

Question 3

The screenshot shows a Snowflake SQL query execution interface. The query is as follows:

```
15  
16 -- Q3. Display all the distinct product categories in the dataset.  
17 -- Expected output: Product Category  
18 SELECT DISTINCT product_category  
19 From retail.sales.retail_sales;  
20
```

The results are displayed in a table format, showing 3 rows and 82ms execution time. The table has a single column named **PRODUCT_CATEGORY** with the following values:

PRODUCT_CATEGORY
1 Clothing
2 Beauty
3 Electronics

Question 4

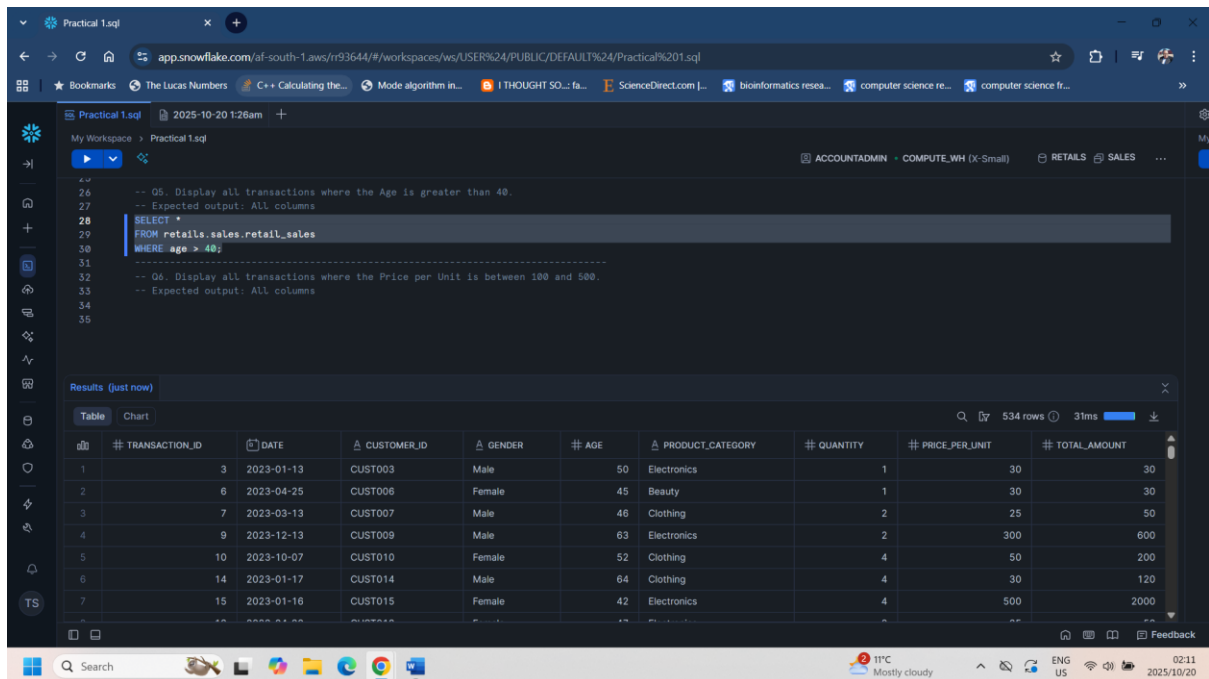
The screenshot shows a Snowflake SQL query execution interface. The query is as follows:

```
21 -- Q4. Display all the distinct gender values in the dataset.  
22 -- Expected output: Gender  
23 SELECT DISTINCT gender  
24 From retail.sales.retail_sales;  
25
```

The results are displayed in a table format, showing 2 rows and 78ms execution time. The table has a single column named **GENDER** with the following values:

GENDER
1 Male
2 Female

Question 5



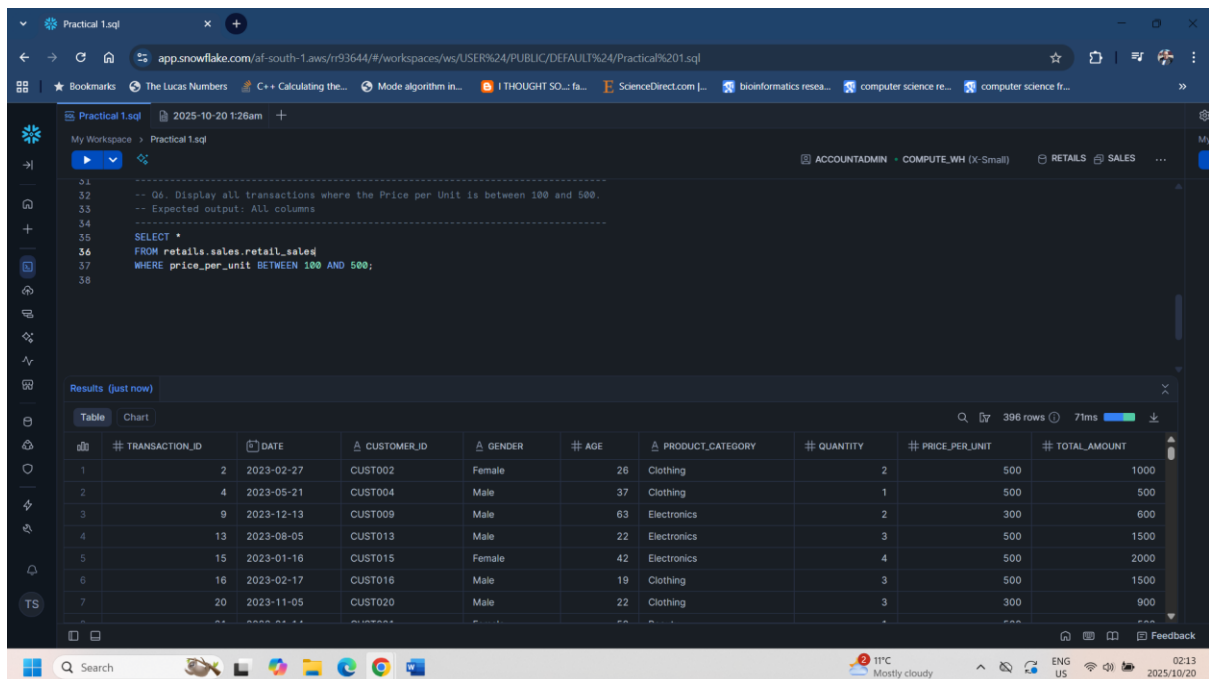
The screenshot shows the Snowflake SQL editor interface. The query being executed is:

```
-- Q5. Display all transactions where the Age is greater than 40.
-- Expected output: All columns
SELECT *
FROM retail_sales.retail_sales
WHERE age > 40;
```

The results table displays the following data:

#	TRANSACTION_ID	DATE	CUSTOMER_ID	GENDER	AGE	PRODUCT_CATEGORY	QUANTITY	PRICE_PER_UNIT	TOTAL_AMOUNT
1	3	2023-01-13	CUST003	Male	50	Electronics	1	30	30
2	6	2023-04-25	CUST006	Female	45	Beauty	1	30	30
3	7	2023-03-13	CUST007	Male	46	Clothing	2	25	50
4	9	2023-12-13	CUST009	Male	63	Electronics	2	300	600
5	10	2023-10-07	CUST010	Female	52	Clothing	4	50	200
6	14	2023-01-17	CUST014	Male	64	Clothing	4	30	120
7	15	2023-01-16	CUST015	Female	42	Electronics	4	500	2000

Question 6



The screenshot shows the Snowflake SQL editor interface. The query being executed is:

```
-- Q6. Display all transactions where the Price per Unit is between 100 and 500.
-- Expected output: All columns
SELECT *
FROM retail_sales.retail_sales
WHERE price_per_unit BETWEEN 100 AND 500;
```

The results table displays 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
6	16	2023-02-17	CUST016	Male	19	Clothing	3	500	1500
7	20	2023-11-05	CUST020	Male	22	Clothing	3	300	900

Question 7

The screenshot shows a Snowflake SQL editor with a query that filters transactions by product category. The query is as follows:

```
38  
39 -- Q7. Display all transactions where the Product Category is either 'Beauty' or 'Electronics'.  
40 -- Expected output: All columns  
41 SELECT *  
42 FROM retail_sales.retail_sales  
43 WHERE product_category = 'Beauty' OR product_category = 'Electronics';  
44 Ctrl+I to generate
```

The results table shows 649 rows and 73ms execution time. 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 filtered to include only transactions where the product category is either 'Beauty' or 'Electronics'.

#	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
6	9	2023-12-13	CUST009	Male	63	Electronics	2	300	600
7	12	2023-10-30	CUST012	Male	35	Beauty	3	25	75
8	13	2023-08-05	CUST013	Male	22	Electronics	3	500	1500
9	15	2023-01-16	CUST015	Female	42	Electronics	4	500	2000
10	18	2023-04-30	CUST018	Female	47	Electronics	2	25	50
11	21	2023-01-14	CUST021	Female	50	Beauty	1	500	500

Question 8

The screenshot shows a Snowflake SQL editor with a query that filters transactions by product category. The query is as follows:

```
45 -- Q8. Display all transactions where the Product Category is not 'Clothing'.  
46 -- Expected output: All columns  
47 SELECT *  
48 FROM retail_sales.retail_sales  
49 WHERE product_category != 'Clothing';  
50  
51
```

The results table shows 649 rows and 69ms execution time. 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 filtered to include only transactions where the product category is not 'Clothing'.

#	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
6	9	2023-12-13	CUST009	Male	63	Electronics	2	300	600
7	12	2023-10-30	CUST012	Male	35	Beauty	3	25	75
8	13	2023-08-05	CUST013	Male	22	Electronics	3	500	1500
9	15	2023-01-16	CUST015	Female	42	Electronics	4	500	2000
10	18	2023-04-30	CUST018	Female	47	Electronics	2	25	50
11	21	2023-01-14	CUST021	Female	50	Beauty	1	500	500

Question 9

The screenshot shows a Snowflake SQL query editor with the following SQL code:

```
50  
51 -- Q9. Display all transactions where the Quantity is greater than or equal to 3.  
52 -- Expected output: All columns  
53 SELECT *  
54 FROM retail.sales.retail_sales  
55 WHERE quantity >= 3;  
56
```

The results are displayed in a table with 10 columns: TRANSACTION_ID, DATE, CUSTOMER_ID, GENDER, AGE, PRODUCT_CATEGORY, QUANTITY, PRICE_PER_UNIT, and TOTAL_AMOUNT. The table contains 12 rows of 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
6	2023-01-17	CUST014	Male	64	Clothing	4	30	120
7	2023-01-16	CUST015	Female	42	Electronics	4	500	2000
8	2023-02-17	CUST016	Male	19	Clothing	3	500	1500
9	2023-04-22	CUST017	Female	27	Clothing	4	25	100
10	2023-11-05	CUST020	Male	22	Clothing	3	300	900
11	2023-04-12	CUST023	Female	35	Clothing	4	30	120
12	2023-10-29	CUST030	Female	39	Beauty	3	300	900

Question 10

The screenshot shows a Snowflake SQL query editor with the following SQL code:

```
56  
57 -- Q10. Count the total number of transactions.  
58 -- Expected output: Total_Transactions  
59 SELECT COUNT(transaction_id) AS total_transactions  
60 FROM retail.sales.retail_sales;
```

The results are displayed in a table with 1 column: TOTAL_TRANSACTIONS. The table contains 1 row of data.

TOTAL_TRANSACTIONS
1000

Question 11

The screenshot shows the Snowflake SQL editor interface. The query being executed is:

```
61 -----
62 -- Q11. Find the average Age of customers.
63 -- Expected output: Average_Age
64 SELECT AVG(age) AS average_age
65 FROM retails.sales.retail_sales;
```

The results pane shows a single row with the value 41.392000.

#	AVERAGE_AGE
1	41.392000

Question 12

The screenshot shows the Snowflake SQL editor interface. The query being executed is:

```
66 -----
67 --Q12. Find the total quantity of products sold.
68 --Expected output: Total_Quantity
69 SELECT SUM(quantity) AS total_quantity
70 FROM retails.sales.retail_sales;
71 -----
```

The results pane shows a single row with the value 2514.

#	TOTAL_QUANTITY
1	2514

Question 13

The screenshot shows a Snowflake workspace with a SQL query executed. The query is as follows:

```
-----  
71  
72 -- Q13. Find the maximum Total Amount spent in a single transaction.  
73 -- Expected output: Max_Total_Amount  
74 SELECT MAX(total_amount) AS max_total_amount  
75 FROM retail.sales.retail_sales;  
76  
77  
-----
```

The results pane shows a single row with the value 2000 for the column MAX_TOTAL_AMOUNT.

MAX_TOTAL_AMOUNT
2000

Question 14

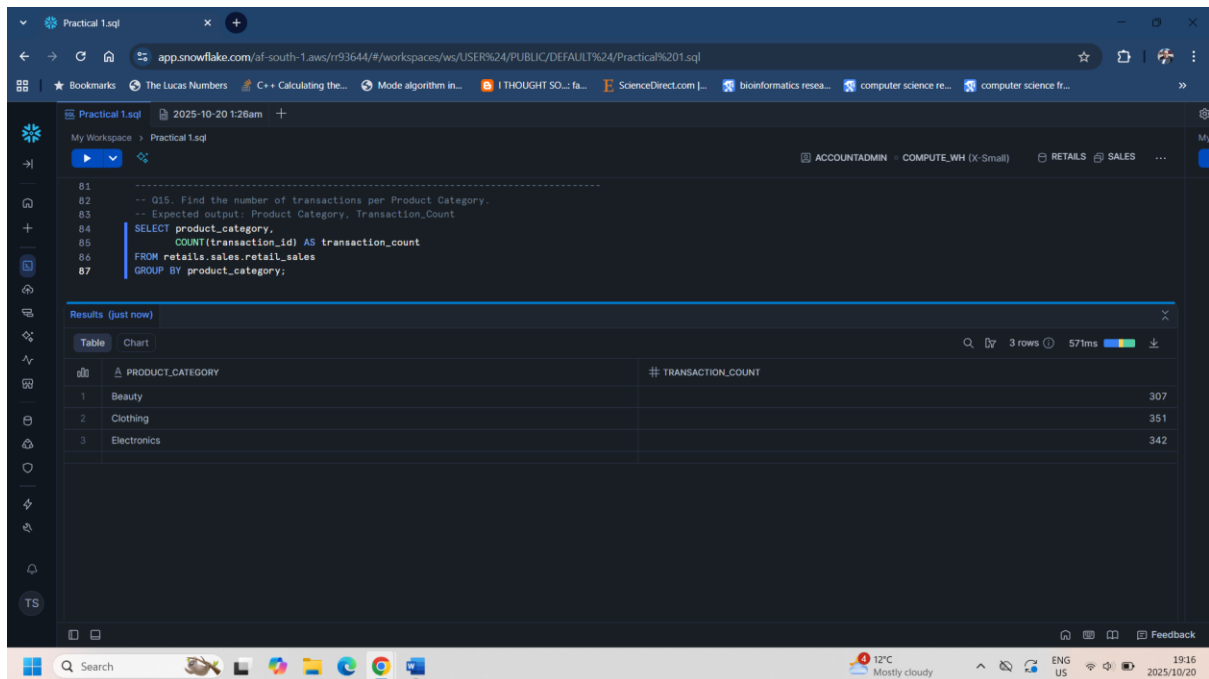
The screenshot shows a Snowflake workspace with a SQL query executed. The query is as follows:

```
-----  
76  
77 -- Q14. Find the minimum Price per Unit in the dataset.  
78 -- Expected output: Min_Price_per_Unit  
79 SELECT MIN(price_per_unit) AS min_price_per_unit  
80 FROM retail.sales.retail_sales;  
81  
-----
```

The results pane shows a single row with the value 25 for the column MIN_PRICE_PER_UNIT.

MIN_PRICE_PER_UNIT
25

Question 15



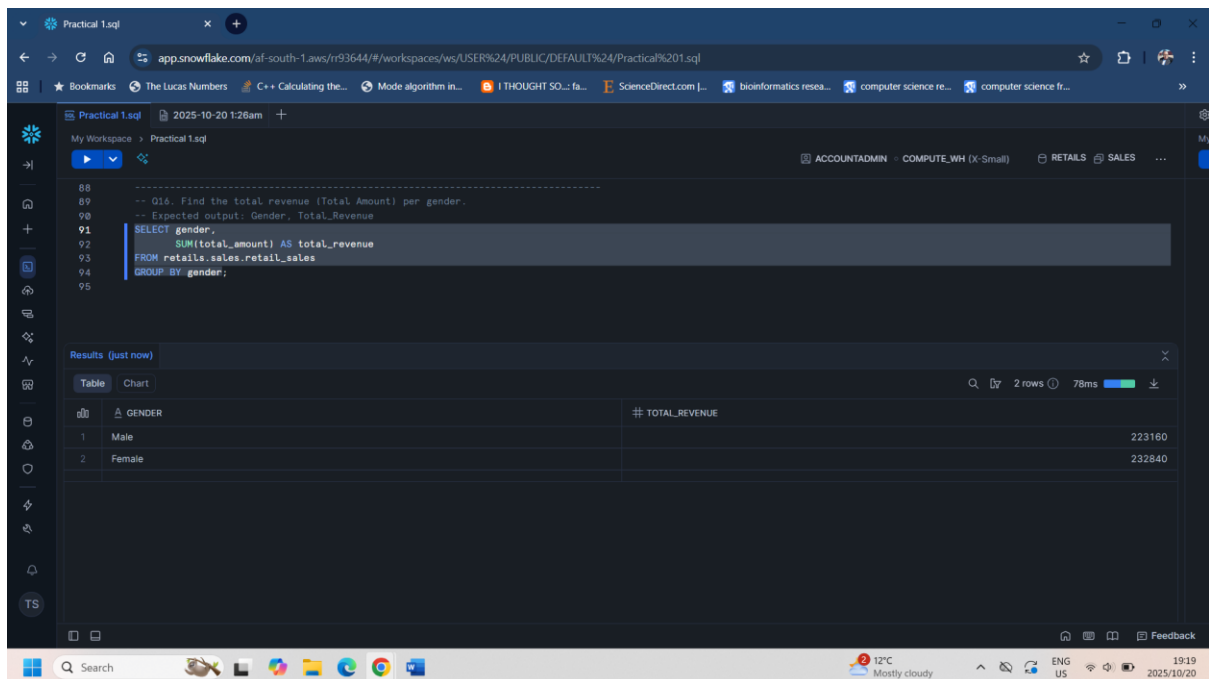
The screenshot shows the Snowflake SQL interface with a query executed. The query is as follows:

```
81 -----
82 -- Q15. Find the number of transactions per Product Category.
83 -- Expected output: Product Category, Transaction_Count
84 SELECT product_category,
85        COUNT(transaction_id) AS transaction_count
86 FROM retail.sales.retail_sales
87 GROUP BY product_category;
```

The results are displayed in a table with 3 rows and 2 columns:

PRODUCT_CATEGORY	TRANSACTION_COUNT
Beauty	307
Clothing	351
Electronics	342

Question 16



The screenshot shows the Snowflake SQL interface with a query executed. The query is as follows:

```
88 -----
89 -- Q16. Find the total revenue (Total Amount) per gender.
90 -- Expected output: Gender, Total_Revenue
91 SELECT gender,
92        SUM(total_amount) AS total_revenue
93 FROM retail.sales.retail_sales
94 GROUP BY gender;
```

The results are displayed in a table with 2 rows and 2 columns:

GENDER	TOTAL_REVENUE
Male	223160
Female	232840

Question 17

The screenshot shows the Snowflake SQL interface with a query editor and results pane. The query is as follows:

```
95 -----
96 -- Q17. Find the average Price per Unit per product category.
97 -- Expected output: Product Category, Average_Price
98 SELECT product_category,
99        AVG(price_per_unit) AS average_price
100 FROM retail_sales.retail_sales
101 GROUP BY product_category;
102
103
```

The results pane shows the following data:

PRODUCT_CATEGORY	AVERAGE_PRICE
1 Beauty	184.055375
2 Clothing	174.287749
3 Electronics	181.900585

Question 18

The screenshot shows the Snowflake SQL interface with a query editor and results pane. The query is as follows:

```
102 -----
103 -- Q18. Find the total revenue per product category where total revenue is greater than
104 -- 10,000
105 -- Expected output: Product Category, Total_Revenue
106 SELECT product_category,
107        SUM(total_amount) AS total_revenue
108 FROM retail_sales.retail_sales
109 GROUP BY product_category
110 HAVING total_revenue > 10,000;
111
```

The results pane shows the following data:

PRODUCT_CATEGORY	TOTAL_REVENUE
1 Beauty	143515
2 Clothing	155580
3 Electronics	156905

Question 19

The screenshot shows the Snowflake SQL editor interface. The query being executed is:

```
-- Q19. Find the average quantity per product category where the average is more than 2.
-- Expected output: Product Category, Average_Quantity
SELECT product_category,
       AVG(quantity) AS average_quantity
FROM retail.sales.retail_sales
GROUP BY product_category
HAVING average_quantity > 2;
```

The results are displayed in a table with 3 rows and 2 columns: PRODUCT_CATEGORY and AVERAGE_QUANTITY.

PRODUCT_CATEGORY	AVERAGE_QUANTITY
Beauty	2.511401
Clothing	2.547009
Electronics	2.482456

Question 20

The screenshot shows the Snowflake SQL editor interface. The query being executed is:

```
-- Q20. Display a column called Spending_Level that shows 'High' if Total Amount > 1000,
-- otherwise 'Low'.
-- Expected output: Transaction ID, Total Amount, Spending_Level
SELECT transaction_id,
       total_amount,
       CASE
         WHEN total_amount > 1000 THEN 'High'
         ELSE 'Low'
       END AS spending_level
FROM retail.sales.retail_sales;
```

The results are displayed in a table with 9 rows and 3 columns: TRANSACTION_ID, TOTAL_AMOUNT, and SPENDING_LEVEL.

TRANSACTION_ID	TOTAL_AMOUNT	SPENDING_LEVEL
1	150	Low
2	1000	Low
3	30	Low
4	500	Low
5	100	Low
6	30	Low
7	50	Low
8	100	Low
9	600	Low

Question 21

The screenshot shows a Snowflake SQL editor interface. The query being executed is a CASE statement that categorizes customers into age groups based on their age. The results table shows 7 rows of data.

```
130
131 -- Q21. Display a new column called Age_Group that labels customers as:
132 -- - 'Youth' if Age < 30
133 -- - 'Adult' if Age is between 30 and 59
134 -- - 'Senior' if Age >= 60
135 -- Expected output: Customer ID, Age, Age_Group
136 SELECT customer_id,
137        age,
138        CASE
139          WHEN age < 30 THEN 'Youth'
140          WHEN age BETWEEN 30 AND 59 THEN 'Adult'
141          WHEN age >= 60 THEN 'Senior'
142        END AS age_group
143 FROM retail.sales.retail_sales;
```

Results (just now)

	CUSTOMER_ID	AGE	AGE_GROUP
1	CUST001	34	Adult
2	CUST002	26	Young
3	CUST003	50	Adult
4	CUST004	37	Adult
5	CUST005	30	Adult
6	CUST006	45	Adult
7	CUST007	48	Adult