

# BrightLight Data Analytics Coding Practical

## Practical 1: SQL Fundamentals (Snowflake-Basic SQL Syntax)

The following questions are designed to help you build a strong foundation in basic SQL syntax. You are provided with a dataset named **retail\_sales\_dataset.csv**. Upload this dataset to your Snowflake account and use it to answer the questions below.

Please follow the instructions below carefully:

1. Write one SQL query per question.  
*Use proper formatting and indentation where necessary.*
2. Each question tests a specific SQL concept.  
*Read the question carefully and apply only the concept being tested (e.g., SELECT, WHERE, GROUP BY, etc.).*
3. Do not combine multiple SQL concepts unless instructed to.  
*For example, if the question asks for a SELECT DISTINCT, don't use WHERE unless specified.*
4. Use the correct column names as provided in the dataset
5. Expected Output Columns are provided for each question.  
*Your query must return exactly those columns in the result.*
6. Don't worry about the actual data.  
*Focus on getting the SQL syntax right.*
7. Submit your completed SQL queries as a .sql

Table 1: Outlines the name and descriptions of the columns in the provided dataset  
**"retail\_sales\_dataset.csv"**

Column Name	Description
Transaction_ID	Unique identifier for each transaction.
Date	The date on which the transaction occurred (format: YYYY-MM-DD).
Customer_ID	Unique identifier for the customer making the purchase.
Gender	Gender of the customer (e.g., Male, Female).
Age	Age of the customer at the time of the transaction.
Product_Category	Category of the product purchased (e.g., Beauty, Clothing, Electronics).
Quantity	Number of product units purchased in the transaction.
Price_per_Unit	Cost of a single unit of the product (in the currency used).
Total_Amount	Total amount spent for the transaction (Quantity × Price per Unit).

# Questions

## 1. SELECT Statement

**Q1.** Display all columns for all transactions.

*Expected output:* All columns

The screenshot shows a database interface with a code editor at the top containing the following SQL query:

```
1 | SELECT
2 | *
3 | FROM
4 | "PROJECT"."PUBLIC"."RETAIL_SALES"
5 | 
```

Below the code editor is a results table with the following data:

	# TRANSACTION_ID	⌚ DATE	▲ CUSTOMER_ID	▲ GENDER	# AGE	▲ PRODUCT_CATEGORY	# QUANTITY
1	1	2023-11-24	CUST001	Male	34	Beauty	3
2	2	2023-02-27	CUST002	Female	26	Clothing	2
3	3	2023-01-13	CUST003	Male	50	Electronics	1
4	4	2023-05-21	CUST004	Male	37	Clothing	1
5	5	2023-05-06	CUST005	Male	30	Beauty	2
6	6	2023-04-25	CUST006	Female	45	Beauty	1
7	7	2023-03-13	CUST007	Male	46	Clothing	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 database interface with a code editor at the top containing the following SQL query:

```
1 | SELECT TRANSACTION_ID, DATE, CUSTOMER_ID
2 | FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 | 
```

Below the code editor is a results table with the following data:

	# 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
8	8	2023-02-22	CUST008

## 2. SELECT DISTINCT Statement

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

*Expected output:* Product Category

The screenshot shows a database interface with a code editor at the top containing the following SQL query:

```
1 | SELECT DISTINCT(PRODUCT_CATEGORY)
2 | FROM"PROJECT"."PUBLIC"."RETAIL_SALES"
3 |
```

Below the code editor is a results table with the following data:

	PRODUCT_CATEGORY
1	Electronics
2	Clothing
3	Beauty

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

*Expected output:* Gender

The screenshot shows a database interface with a code editor at the top containing the following SQL query:

```
1 | SELECT DISTINCT(GENDER)
2 | FROM"PROJECT"."PUBLIC"."RETAIL_SALES"
3 |
```

Below the code editor is a results table with the following data:

	GENDER
1	Male
2	Female

---

## 3. WHERE Clause

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

*Expected output:* All columns

PROJECT.PUBLIC ▾ Settings ▾

Code Versions

```

1 | SELECT *
2 | FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 | WHERE AGE > 40
4 |

```

↳ Results

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

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

*Expected output:* All columns

PROJECT.PUBLIC ▾ Settings ▾

Code Versions

```

1 | SELECT *
2 | FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 | WHERE PRICE_PER_UNIT BETWEEN 100 AND 500
4 |

```

↳ Results

⌚ DATE	▲ CUSTOMER_ID	▲ GENDER	# AGE	▲ PRODUCT_CATEGORY	# QUANTITY	# PRICE_PER_UNIT
1 2023-02-27	CUST002	Female	26	Clothing	2	500
2 2023-05-21	CUST004	Male	37	Clothing	1	500
3 2023-12-13	CUST009	Male	63	Electronics	2	300
4 2023-08-05	CUST013	Male	22	Electronics	3	500
5 2023-01-16	CUST015	Female	42	Electronics	4	500
6 2023-02-17	CUST016	Male	19	Clothing	3	500
7 2023-11-05	CUST020	Male	22	Clothing	3	300

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

*Expected output:* All columns

PROJECT.PUBLIC ▾ Settings ▾ Code Versions ▾

```
1 SELECT *
2 FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 WHERE PRODUCT_CATEGORY IN('Beauty', 'Electronics')
4
```

↳ Results ▾ Chart

	# TRANSACTION_ID	⌚ DATE	▲ CUSTOMER_ID	▲ GENDER	# AGE	▲ PRODUCT_CATEGORY	# QUANTITY
1	1	2023-11-24	CUST001	Male	34	Beauty	3
2	3	2023-01-13	CUST003	Male	50	Electronics	1
3	5	2023-05-06	CUST005	Male	30	Beauty	2
4	6	2023-04-25	CUST006	Female	45	Beauty	1
5	8	2023-02-22	CUST008	Male	30	Electronics	4
6	9	2023-12-13	CUST009	Male	63	Electronics	2
7	12	2023-10-30	CUST012	Male	35	Beauty	3

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

*Expected output:* All columns

PROJECT.PUBLIC ▾ Settings ▾ Code Versions ▾

```
1 SELECT *
2 FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 WHERE PRODUCT_CATEGORY != 'Clothing'
4
```

↳ Results ▾ Chart

	# TRANSACTION_ID	⌚ DATE	▲ CUSTOMER_ID	▲ GENDER	# AGE	▲ PRODUCT_CATEGORY	# QUANTITY
1	1	2023-11-24	CUST001	Male	34	Beauty	3
2	3	2023-01-13	CUST003	Male	50	Electronics	1
3	5	2023-05-06	CUST005	Male	30	Beauty	2
4	6	2023-04-25	CUST006	Female	45	Beauty	1
5	8	2023-02-22	CUST008	Male	30	Electronics	4
6	9	2023-12-13	CUST009	Male	63	Electronics	2
7	12	2023-10-30	CUST012	Male	35	Beauty	3

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

*Expected output:* All columns

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

```
1 | SELECT *
2 | FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 | WHERE QUANTITY >= 3
4 |
```

The results table has the following data:

	# TRANSACTION_ID	(L) DATE	A CUSTOMER_ID	A GENDER	# AGE	A PRODUCT_CATEGORY	# QUANTITY
1	1	2023-11-24	CUST001	Male	34	Beauty	3
2	8	2023-02-22	CUST008	Male	30	Electronics	4
3	10	2023-10-07	CUST010	Female	52	Clothing	4
4	12	2023-10-30	CUST012	Male	35	Beauty	3
5	13	2023-08-05	CUST013	Male	22	Electronics	3
6	14	2023-01-17	CUST014	Male	64	Clothing	4
7	15	2023-01-16	CUST015	Female	42	Electronics	4

#### 4. Aggregate Functions

**Q10.** Count the total number of transactions.

*Expected output:* Total\_Transactions

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

```
1 | SELECT COUNT(*) AS TOTAL_TRANSACTIONS
2 | FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 |
```

The results table has the following data:

	# TOTAL_TRANSACTIONS
1	1000

**Q11.** Find the average Age of customers.

*Expected output:* Average\_Age

The screenshot shows a database interface with a dark theme. At the top, it displays "PROJECT.PUBLIC" and "Settings" dropdowns, along with "Code Versions" and a search icon. The main area contains a SQL query:

```
1 | SELECT AVG(AGE) AS AVERAGE_AGE
2 | FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 |
```

Below the query, there are two tabs: "Results" (which is selected and highlighted in blue) and "Chart". The results table has one row with the following data:

	AVERAGE_AGE
1	41.392000

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

*Expected output:* Total\_Quantity

The screenshot shows a database interface with a dark theme. At the top, it displays "PROJECT.PUBLIC" and "Settings" dropdowns, along with "Code Versions" and a search icon. The main area contains a SQL query:

```
1 | SELECT SUM(QUANTITY) AS TOTAL_QUANTITY
2 | FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 |
```

Below the query, there are two tabs: "Results" (selected) and "Chart". The results table has one row with the following data:

	TOTAL_QUANTITY
1	2514

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

*Expected output:* Max\_Total\_Amount

The screenshot shows a database interface with a dark theme. At the top, it displays 'PROJECT.PUBLIC' and 'Settings'. On the right, there are 'Code Versions' and a search icon. The main area contains a code editor with the following SQL query:

```
1 | SELECT MAX(TOTAL_AMOUNT) AS MAX_TOTAL_AMOUNT
2 | FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 |
```

Below the code editor is a results table with two columns: '# MAX\_TOTAL\_AMOUNT'. The table has one row with the value '2000'. There are also 'Results' and 'Chart' tabs at the bottom of the results section.

#	MAX_TOTAL_AMOUNT
1	2000

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

*Expected output:* Min\_Price\_per\_Unit

The screenshot shows a database interface with a dark theme. At the top, it displays 'PROJECT.PUBLIC' and 'Settings'. On the right, there are 'Code Versions' and a search icon. The main area contains a code editor with the following SQL query:

```
1 | SELECT MIN(PRICE_PER_UNIT) AS MIN_PRICE_PER_UNIT
2 | FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 |
```

Below the code editor is a results table with two columns: '# MIN\_PRICE\_PER\_UNIT'. The table has one row with the value '25'. There are also 'Results' and 'Chart' tabs at the bottom of the results section.

#	MIN_PRICE_PER_UNIT
1	25

---

## 5. GROUP BY Statement

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

*Expected output:* Product Category, Transaction\_Count

PROJECT.PUBLIC ▾ Settings ▾ Code Versions ⌂

```

1 | SELECT PRODUCT_CATEGORY, COUNT(TRANSACTION_ID) AS TRANSACTION_COUNT
2 | FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 | GROUP BY PRODUCT_CATEGORY
4 |

```

↳ Results ↵ Chart ⌂ ⌂ ⌂ ⌂ ⌂

	PRODUCT_CATEGORY	TRANSACTION_COUNT
1	Clothing	351
2	Beauty	307
3	Electronics	342

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

*Expected output:* Gender, Total\_Revenue

PROJECT.PUBLIC ▾ Settings ▾ Code Versions ⌂

```

1 | SELECT GENDER, SUM(TOTAL_AMOUNT) AS TOTAL_REVENUE
2 | FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 | GROUP BY GENDER

```

↳ Results ↵ Chart ⌂ ⌂ ⌂ ⌂ ⌂

	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

PROJECT.PUBLIC ▾ Settings ▾ Code Versions ⌂

```

1 | SELECT PRODUCT_CATEGORY, AVG(PRICE_PER_UNIT) AS AVERAGE_PRICE
2 | FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 | GROUP BY PRODUCT_CATEGORY

```

↳ Results ↵ Chart ⌂ ⌂ ⌂ ⌂ ⌂

	PRODUCT_CATEGORY	AVERAGE_PRICE
1	Clothing	174.287749
2	Beauty	184.055375
3	Electronics	181.900585

## 6. HAVING Clause

**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 database interface with a dark theme. At the top, there are tabs for 'PROJECT.PUBLIC' and 'Settings'. On the right, there are buttons for 'Code Versions' and a magnifying glass icon. Below the tabs, a SQL query is displayed:

```
1 | SELECT PRODUCT_CATEGORY, SUM(TOTAL_AMOUNT) AS TOTAL_REVENUE
2 | FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 | GROUP BY PRODUCT_CATEGORY
4 | HAVING SUM(TOTAL_AMOUNT) > 10000
```

Below the query, there are two buttons: 'Results' (highlighted in blue) and 'Chart'. To the right of these buttons are icons for search, refresh, and other operations. The results table has two columns: 'PRODUCT\_CATEGORY' and 'TOTAL\_REVENUE'. The data is as follows:

	PRODUCT_CATEGORY	TOTAL_REVENUE
1	Clothing	155580
2	Beauty	143515
3	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 database interface with a dark theme. At the top, there are tabs for 'PROJECT.PUBLIC' and 'Settings'. On the right, there are buttons for 'Code Versions' and a magnifying glass icon. Below the tabs, a SQL query is displayed:

```
1 | SELECT PRODUCT_CATEGORY, AVG(QUANTITY) AS AVERAGE_QUANTITY
2 | FROM "PROJECT"."PUBLIC"."RETAIL_SALES"
3 | GROUP BY PRODUCT_CATEGORY
4 | HAVING AVG(QUANTITY) > 2
```

Below the query, there are two buttons: 'Results' (highlighted in blue) and 'Chart'. To the right of these buttons are icons for search, refresh, and other operations. The results table has two columns: 'PRODUCT\_CATEGORY' and 'AVERAGE\_QUANTITY'. The data is as follows:

	PRODUCT_CATEGORY	AVERAGE_QUANTITY
1	Clothing	2.547009
2	Beauty	2.511401
3	Electronics	2.482456

---

## 7. CASE Statement

**Q20.** Display a column called Spending\_Level that shows 'High' if Total Amount > 1000, otherwise 'Low'.

*Expected output:* Transaction ID, Total Amount, Spending\_Level

PROJECT.PUBLIC ▾      Settings ▾      Code Versions      Q

```

1  SELECT TRANSACTION_ID, TOTAL_AMOUNT,
2      CASE
3          WHEN TOTAL_AMOUNT > 1000 THEN 'High'
4          ELSE 'Low'
5      END AS SPENDING_LEVEL
6  FROM "PROJECT"."PUBLIC"."RETAIL_SALES"

```

↳ Results    ↳ Chart

	# TRANSACTION_ID	# TOTAL_AMOUNT	▲ SPENDING_LEVEL
1	1	150	Low
2	2	1000	Low
3	3	30	Low
4	4	500	Low
5	5	100	Low
6	6	30	Low
7	7	50	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

PROJECT.PUBLIC ▾      Settings ▾      Code Versions

```

1  SELECT CUSTOMER_ID, AGE,
2      CASE
3          WHEN AGE < 30 THEN 'Youth'
4          WHEN AGE BETWEEN 30 AND 59 THEN 'Adult'
5          ELSE 'Senior'
6      END AS AGE_GROUP
7  FROM "PROJECT"."PUBLIC"."RETAIL_SALES"

```

↳ Results    ↳ Chart

	▲ 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
6	CUST006	45	Adult