# Your First Queries

November 4, 2025

## 1 Ungraded Lab : Your First Queries with BookCycle

### 1.1 Overview

As a data analyst at BookCycle, you're tasked with helping the management team understand their customer base better. Building on your previous experience connecting to the database, you'll now write queries to analyze customer data across their three Seattle locations.

### 1.2 Business Context

BookCycle needs to:

- Identify customer distribution across different locations
- Analyze purchase patterns
- Find potential opportunities for targeted marketing

### 1.3 Activities

#### 1.3.1 Activity 1: Database Connection and Initial Exploration

Step 1: First, let's set up our environment and see what our customers table looks like :

[1]:
```python
import sqlite3
import pandas as pd

# Setting up the database. DO NOT edit the code given below
from db_setup import setup_database
setup_database()

# Connect to the database
conn = sqlite3.connect('bookcycle.db')
```

```
Database setup complete: Tables created and populated with data!
```

[2]:
```python
# Example: View the first few customers
query = """
SELECT *
FROM customer
```

```
LIMIT 5;"""

df = pd.read_sql_query(query, conn)
display(df)
```

```
  customer_id   join_date  is_member  zip_code  birth_year preferred_store
0       C1001  2022-01-15          1     98105        1995      University
1       C1002  2022-01-15          0     98115        1988        Suburban
2       C1003  2022-01-16          1     98101        1992        Downtown
3       C1004  2022-01-16          1     98105        1999      University
4       C1005  2022-01-16          0     98115        1975        Suburban
```

Step 2: Try yourself: Write a query to view ALL customers in the database:

```
[3]: query = """
select * from customers;
"""
df = pd.read_sql_query(query, conn)
display(df)
```

```
   customer_id   join_date  is_member  zip_code  birth_year preferred_store
0        C1001  2022-01-15          1     98105        1995      University
1        C1002  2022-01-15          0     98115        1988        Suburban
2        C1003  2022-01-16          1     98101        1992        Downtown
3        C1004  2022-01-16          1     98105        1999      University
4        C1005  2022-01-16          0     98115        1975        Suburban
..         …           …        …       …          …             …
95       C1096  2022-02-16          1     98105        1996      University
96       C1097  2022-02-16          0     98115        1969        Suburban
97       C1098  2022-02-16          1     98101        1987        Downtown
98       C1099  2022-02-17          1     98105        2000      University
99       C1100  2022-02-17          0     98115        1974        Suburban

[100 rows x 6 columns]
```

### 1.3.2 Activity 2 : Basic Data Retrieval

Let's see how to select specific columns. Here's an example where we look into customer information

Step 1: Write a query to view customer IDs, member status and preferred store

```
[4]: query = """
SELECT customer_id,is_member,preferred_store
FROM customer;
"""
df = pd.read_sql_query(query, conn)
display(df)
```

```
   customer_id  is_member preferred_store
```

```
0        C1001         1     University
1        C1002         0       Suburban
2        C1003         1       Downtown
3        C1004         1     University
4        C1005         0       Suburban
..         …         …           …
95       C1096         1     University
96       C1097         0       Suburban
97       C1098         1       Downtown
98       C1099         1     University
99       C1100         0       Suburban

[100 rows x 3 columns]
```

Step 2: Try Yourself: Write a query to show customer_id, join_date, and is_member

[6]:
```
query = """
select customer_id, join_date, is_member
from customer;
"""

df = pd.read_sql_query(query, conn)
display(df)
```

```
   customer_id   join_date  is_member
0        C1001  2022-01-15          1
1        C1002  2022-01-15          0
2        C1003  2022-01-16          1
3        C1004  2022-01-16          1
4        C1005  2022-01-16          0
..         …          …         …
95       C1096  2022-02-16          1
96       C1097  2022-02-16          0
97       C1098  2022-02-16          1
98       C1099  2022-02-17          1
99       C1100  2022-02-17          0

[100 rows x 3 columns]
```

### 1.3.3 Activity 3: Filtering with WHERE

Find specific customers based on conditions.

Step 1: Find customers who prefer the "Downtown" store

[7]:
```
query = """
SELECT customer_id, preferred_store
FROM customer
WHERE preferred_store = 'Downtown';
```

```
"""
df = pd.read_sql_query(query, conn)
display(df)
```

```
   customer_id preferred_store
0        C1003        Downtown
1        C1007        Downtown
2        C1010        Downtown
3        C1014        Downtown
4        C1018        Downtown
5        C1022        Downtown
6        C1025        Downtown
7        C1029        Downtown
8        C1032        Downtown
9        C1036        Downtown
10       C1040        Downtown
11       C1043        Downtown
12       C1047        Downtown
13       C1050        Downtown
14       C1053        Downtown
15       C1056        Downtown
16       C1059        Downtown
17       C1062        Downtown
18       C1065        Downtown
19       C1068        Downtown
20       C1071        Downtown
21       C1074        Downtown
22       C1077        Downtown
23       C1080        Downtown
24       C1083        Downtown
25       C1086        Downtown
26       C1089        Downtown
27       C1092        Downtown
28       C1095        Downtown
29       C1098        Downtown
```

Step 2: Try Yourself: Write a query to find customers who prefer the "University" store

[8]:
```
query = """
select customer_id, preferred_store
from customer
where preferred_store = 'University';
"""
df = pd.read_sql_query(query, conn)
display(df)
```

```
   customer_id preferred_store
0        C1001      University
```

```
1         C1004      University
2         C1006      University
3         C1008      University
4         C1011      University
5         C1012      University
6         C1015      University
7         C1016      University
8         C1019      University
9         C1021      University
10        C1023      University
11        C1026      University
12        C1028      University
13        C1030      University
14        C1033      University
15        C1034      University
16        C1037      University
17        C1039      University
18        C1041      University
19        C1044      University
20        C1046      University
21        C1048      University
22        C1051      University
23        C1054      University
24        C1057      University
25        C1060      University
26        C1063      University
27        C1066      University
28        C1069      University
29        C1072      University
30        C1075      University
31        C1078      University
32        C1081      University
33        C1084      University
34        C1087      University
35        C1090      University
36        C1093      University
37        C1096      University
38        C1099      University
```

### 1.3.4 Activity 4: Using Logical Operators

Let's filter with multiple conditions.

Step 1: Find customers who prefer "Downtown" AND are members

[9]:
```
query = """
SELECT customer_id, is_member, preferred_store
FROM customer
```

```
WHERE preferred_store = 'Downtown'
AND is_member = 1;
"""
df = pd.read_sql_query(query, conn)
display(df)
```

```
   customer_id  is_member preferred_store
0        C1003          1        Downtown
1        C1010          1        Downtown
2        C1014          1        Downtown
3        C1018          1        Downtown
4        C1025          1        Downtown
5        C1032          1        Downtown
6        C1036          1        Downtown
7        C1043          1        Downtown
8        C1050          1        Downtown
9        C1053          1        Downtown
10       C1056          1        Downtown
11       C1062          1        Downtown
12       C1065          1        Downtown
13       C1071          1        Downtown
14       C1074          1        Downtown
15       C1080          1        Downtown
16       C1083          1        Downtown
17       C1089          1        Downtown
18       C1092          1        Downtown
19       C1098          1        Downtown
```

Step 2: Try Yourself: Find customers who prefer either "Downtown" OR "University" stores

[10]:
```
query = """
select customer_id, preferred_store
from customer
where preferred_store = 'Downtown'
or preferred_store = 'University';
"""
df = pd.read_sql_query(query, conn)
display(df)
```

```
   customer_id preferred_store
0        C1001      University
1        C1003        Downtown
2        C1004      University
3        C1006      University
4        C1007        Downtown
..         …               …
64       C1093      University
65       C1095        Downtown
```

```
66       C1096      University
67       C1098        Downtown
68       C1099      University

[69 rows x 2 columns]
```

### 1.3.5 Close the Connection

It's good practice to close the database connection when you're done

[11]:
```
# Close the database connection
conn.close()
```

## 1.4 Success Checklist

After each query, check:

- Does the output match the expected format?
- Are the results logical for BookCycle's business?
- Do the numbers make sense?

## 1.5 Common Issues & Solutions

- Problem: Syntax errors with quotation marks
  - Solution: Always use single quotes (' ') for string values in SQL queries, not double quotes (" ")
- Problem: Missing semicolons at query ends
  - Solution: Add semicolon (;) at the end of each SQL query

## 1.6 Summary

In this lab, you've learned to interact with a real-world bookstore database using SQL in Jupyter Notebook. You've practiced writing queries to retrieve and filter data, essential skills for data analysis and business intelligence. ### Key Points - SQL queries are the foundation for extracting specific data from databases using SELECT, FROM, and WHERE clauses - Filtering and combining conditions with WHERE, AND, and OR allows for precise data selection based on multiple criteria - Data validation and result checking are crucial steps in ensuring query accuracy and meaningful business insights