Wilbur Wildcat
INFO 579: SQL/NoSQL
Summer 2024
Week 7 Final Project Report
Nayem Rahman, PhD
August 16, 2024

Business Plan

1. Come up with a business plan for which you need to collect and store data

The `tackle` database is a central component of the notional 'Fishing Companion' app, designed to enhance the angling experience by organizing and managing comprehensive data on baits, rods, reels, and lines. This database is meticulously structured to store detailed information about various tackle configurations, enabling the retrieval of personalized recommendations based on specific fishing conditions. Users can access the most suitable tackle configurations by inputting details such as water type, target species, and weather conditions. The database's robust design ensures that all recommendations are accurate, up-to-date, and tailored to the unique needs of each user. This project underscores the critical importance of the tackle database in storing, managing, and analyzing data, providing precise and valuable insights that are integral to the project's success. The project repository is at https://github.com/ciioprof0/tackle.

Data Sources

2. Come up with the data about your business.

The data source is a comprehensive dataset that organizes critical information about various fishing tackle components, including baits, rods, reels, and lines. The dataset is divided into multiple sheets, each focusing on a specific tackle type. The **Tackle** sheet consolidates details such as bait type, presentation method, fish depth, rod type and length, reel gear ratio, and line test strength. The **Baits** sheet provides data on bait types, their effective depth range, and presentation methods. The **Rods** sheet contains information on rod power, action, and length specifications. The **Reels** sheet includes data on reel types and gear ratios, while the **Lines** sheet captures the range of line test strengths across different line types. This structured data, derived from tournament fishing experience, is essential for the Fishing Companion app's database, enabling it to offer accurate and practical recommendations to users. The raw data used to seed the database is at https://github.com/ciioprof0/tackle/raw/main/data/raw/data.xlsx.

Data Models

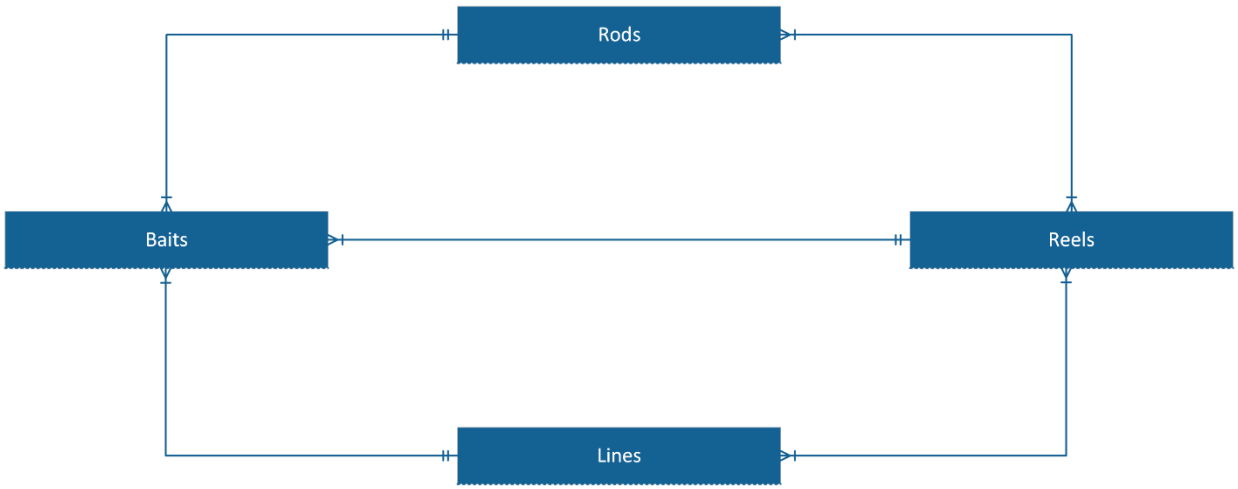
In the relationship between rods and reels, a single rod can be paired with multiple reels, allowing for different combinations of fishing setups. Conversely, a single reel can be used with various rods, offering versatility in utilizing the equipment. Similarly, when considering rods and lines, a single rod can accommodate multiple types of lines and a single line can be applied across different rods, demonstrating a flexible usage pattern. The same principle applies to rods and baits, where a single rod can be used with multiple baits and a single bait can be matched with different rods to suit varying fishing conditions. The `Combos` table is the junction point

that facilitates these many-to-many relationships, linking rods, reels, lines, and baits together in numerous combinations, reflecting the complex interconnections within the fishing gear setup. See Figure 1 for the conceptual model, Figure 2 for the logical model, and Figure 3 for the physical model.

3. Develop a Conceptual Model. Consider 4 or 5 entities. Make sure you have at least one many-to-many relationship. Explain with data why it's a many-to-many relationship.

Figure 1

Tackle Database Conceptual Model with Many-to-Many Relationships

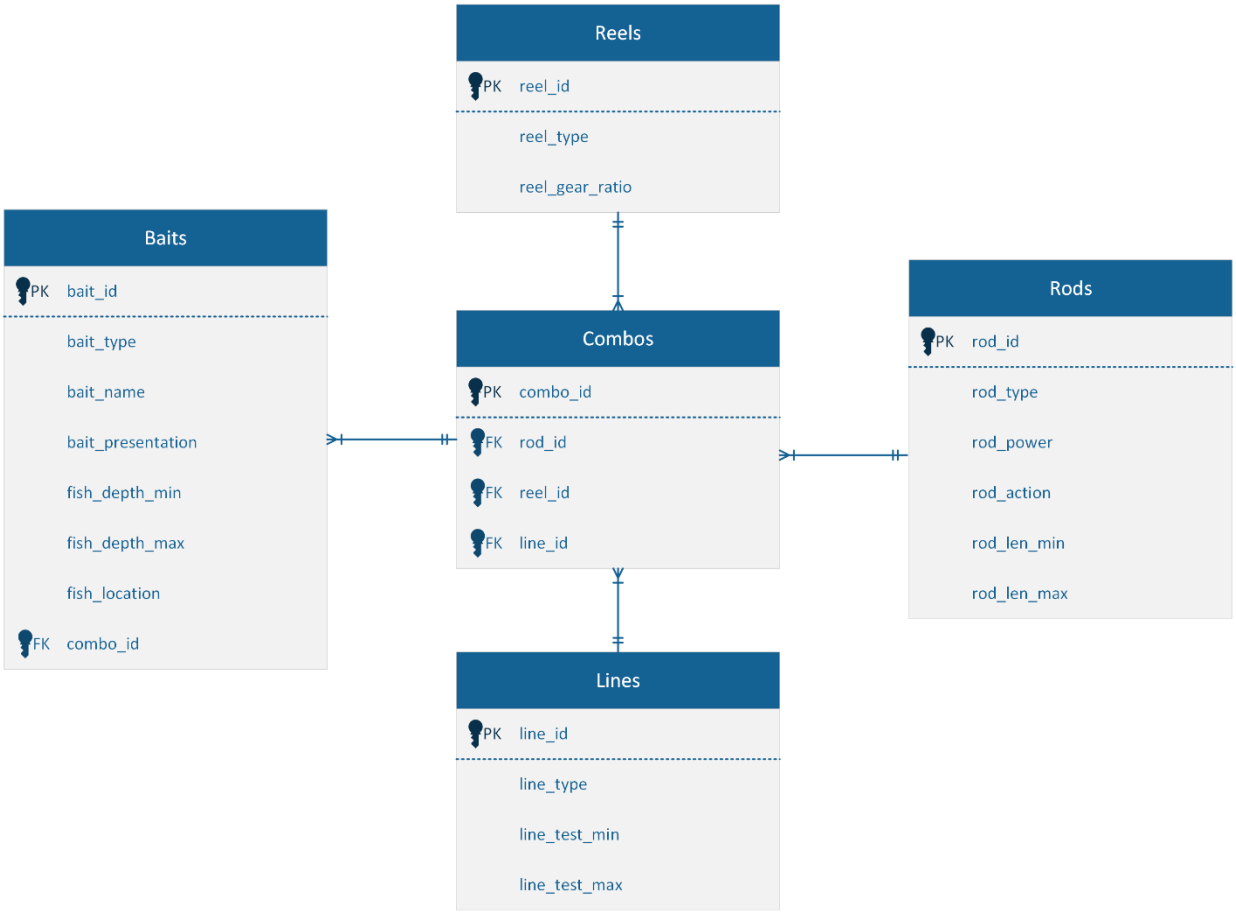


Note. Diagram created with Microsoft Visio 365 for the web. Diagram Visio (vsdx), PDF, and PNG files available at https://github.com/ciioprof0/tackle/tree/main/docs/design

4. Develop a Logical Model using the Conceptual Model. Make sure you come up with a junction entity to resolve the many-to-many relationship.

Figure 2

Tackle Database Logical Model

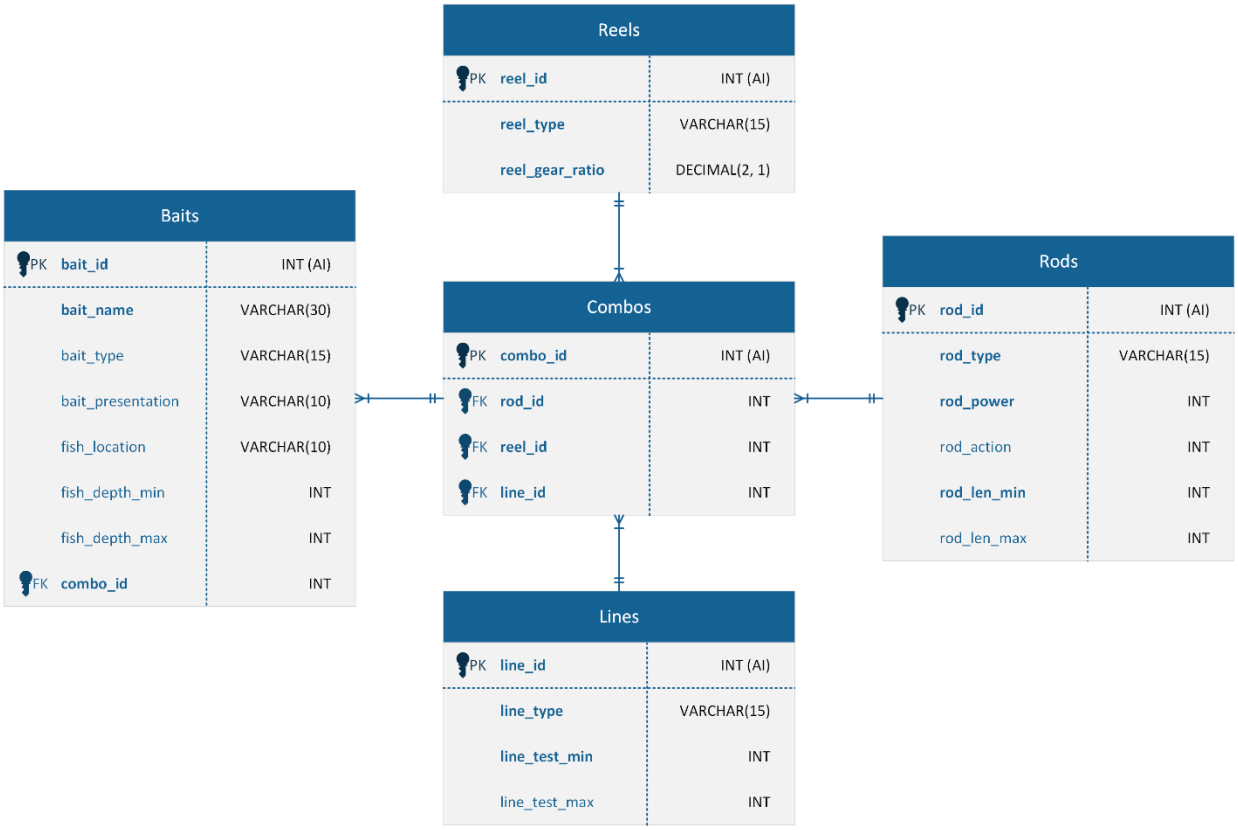


Note. Diagram created with Microsoft Visio 365 for the web. Diagram Visio (vsdx), PDF, and PNG files available at https://github.com/ciioprof0/tackle/tree/main/docs/design

5. Develop the physical model based on the Logical Model

Figure 3

Tackle Database Physical Model



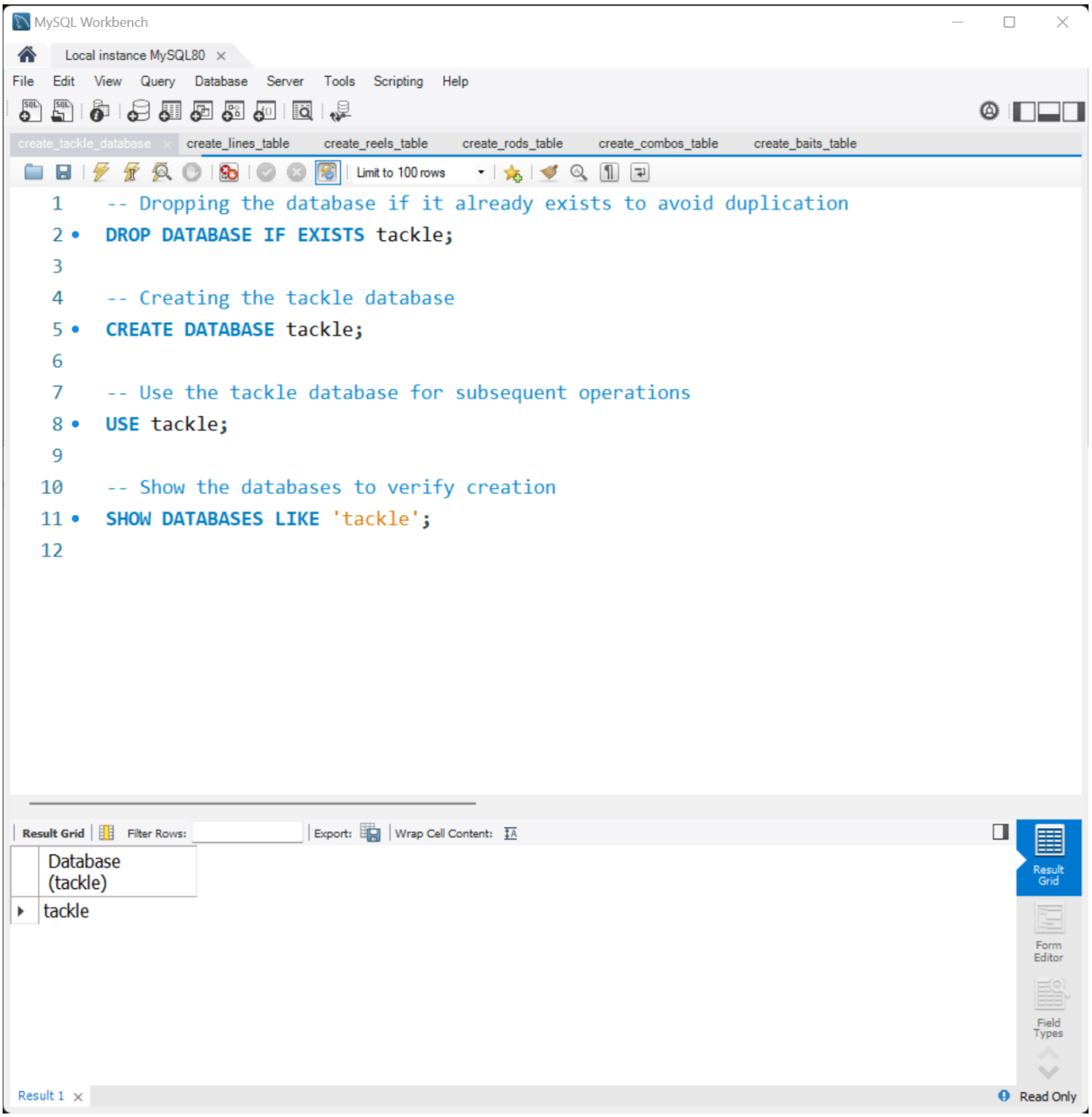
Note. `NOT NULL` attributes are shown in bold font. Diagram created with Microsoft Visio 365 for the web. Diagram Visio (vsdx), PDF, and PNG files available at https://github.com/ciioprof0/tackle/tree/main/docs/design

Schema

Create the database.

Figure 4

Create the Tackle Database



Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/setup/create_tackle_database.sql

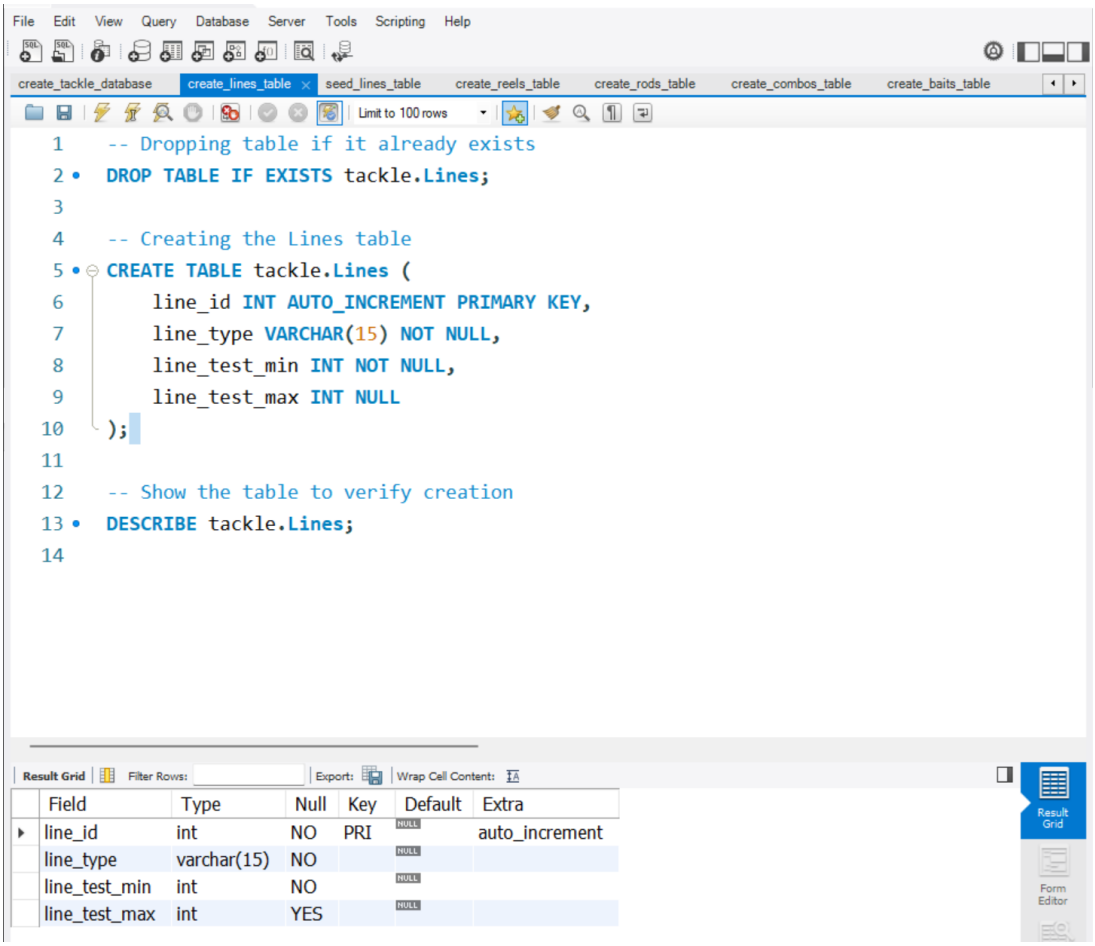
6. Create tables using a database system. Insert data into the database tables. You must provide the DDL (CREATE TABLE statements), INSERT statements, and SELECT statements.

Details: Create the tables that you have come up with (the table must be based on the Physical Model).

- ***(a) Columns, Primary Key (PK), Data Type and length, and NULL/NOT NULL need to be implemented, per the Physical Model.***
- ***(b) Show the table definition (DDL) that you implemented (not in a graphical view).***
- ***(c) Insert the complete set of data that you have come up with and show the insert statements used.***

Figure 5

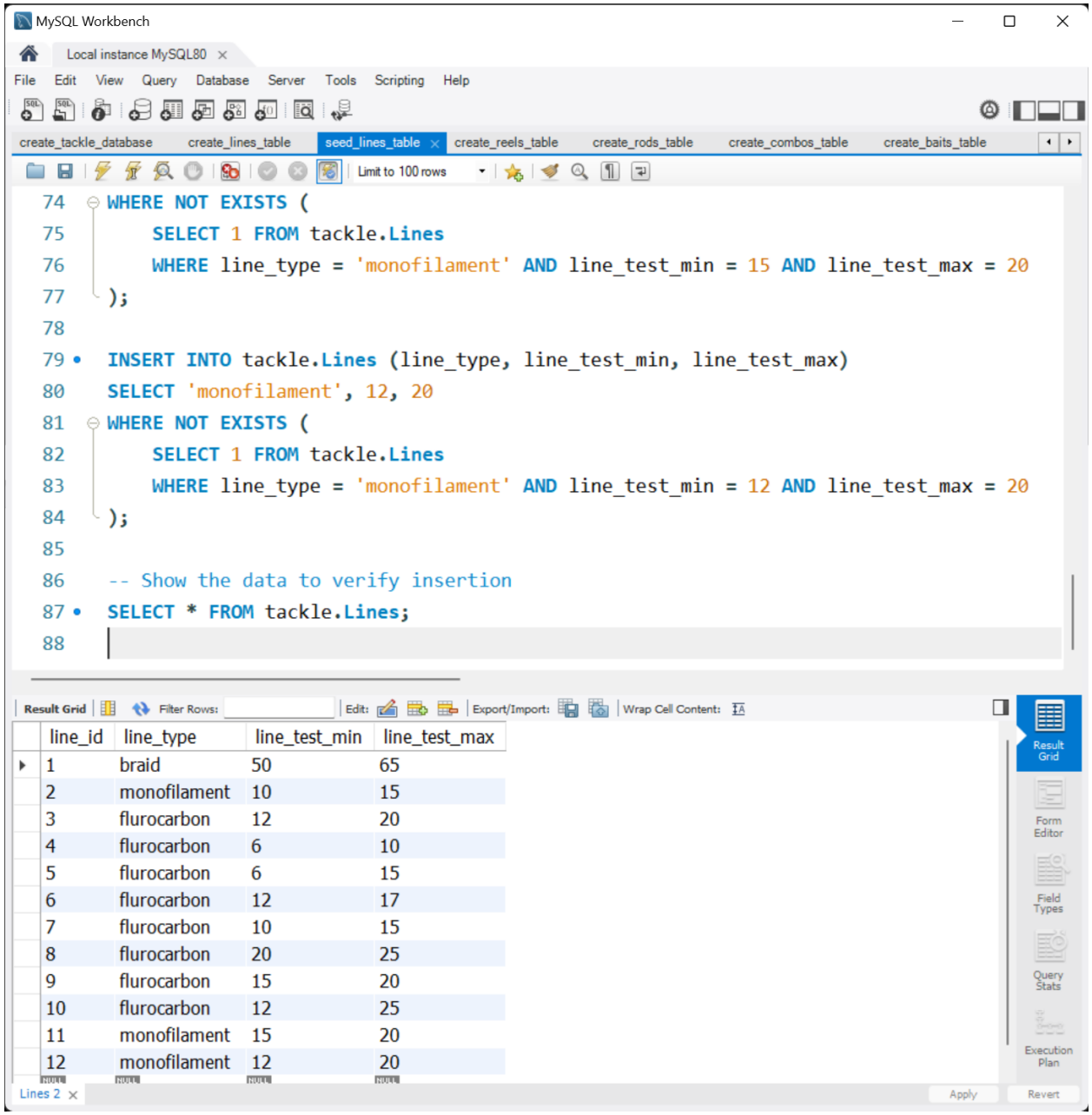
Create Lines Table



Note. Code at
https://github.com/ciioprof0/tackle/blob/main/db/schema/tables/create_lines_table.sql

Figure 6

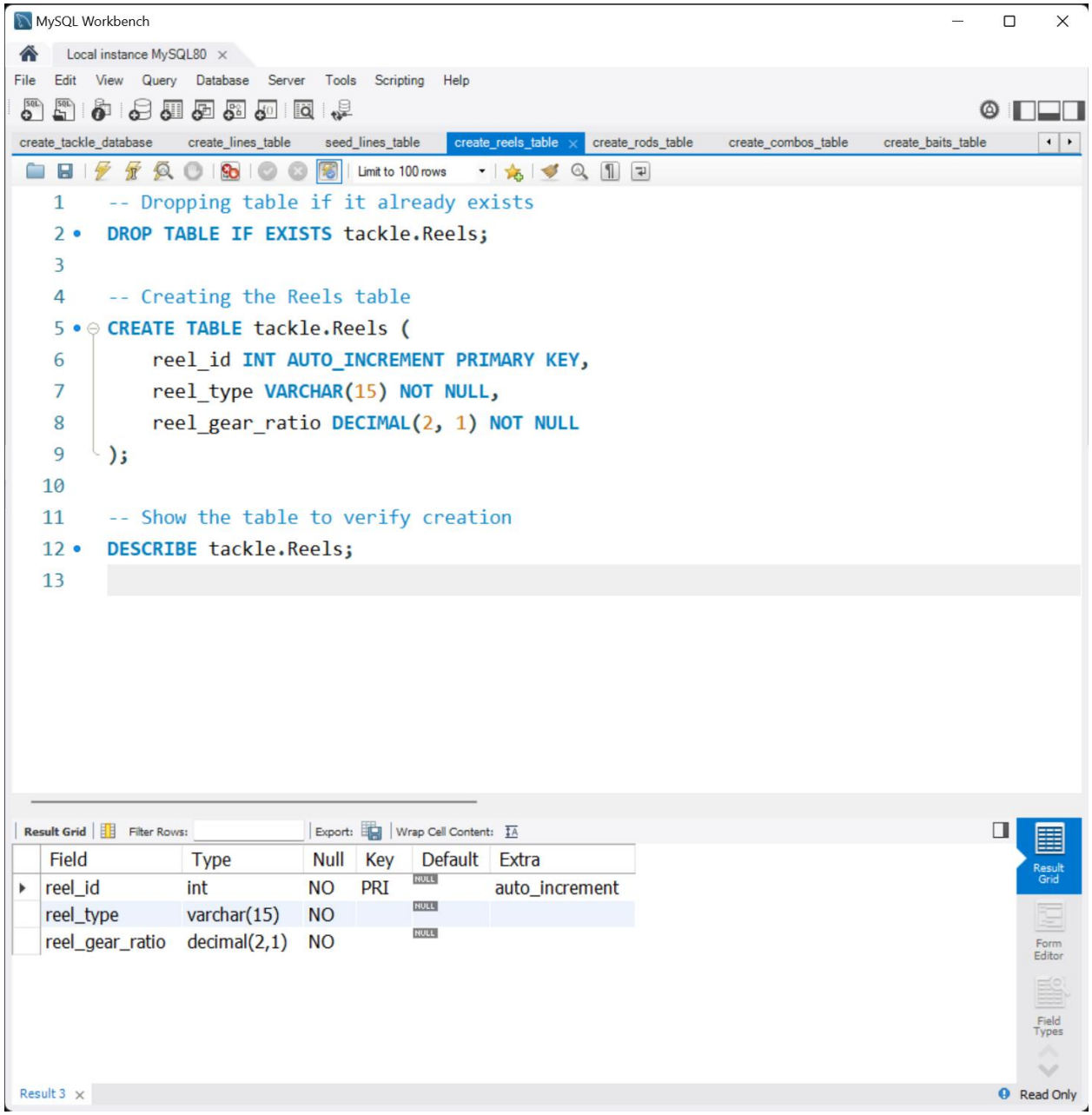
Seed Lines Table



Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/seeds/seed_lines_table.sql
Seed data at https://github.com/ciioprof0/tackle/raw/main/data/raw/data.xlsx

Figure 7

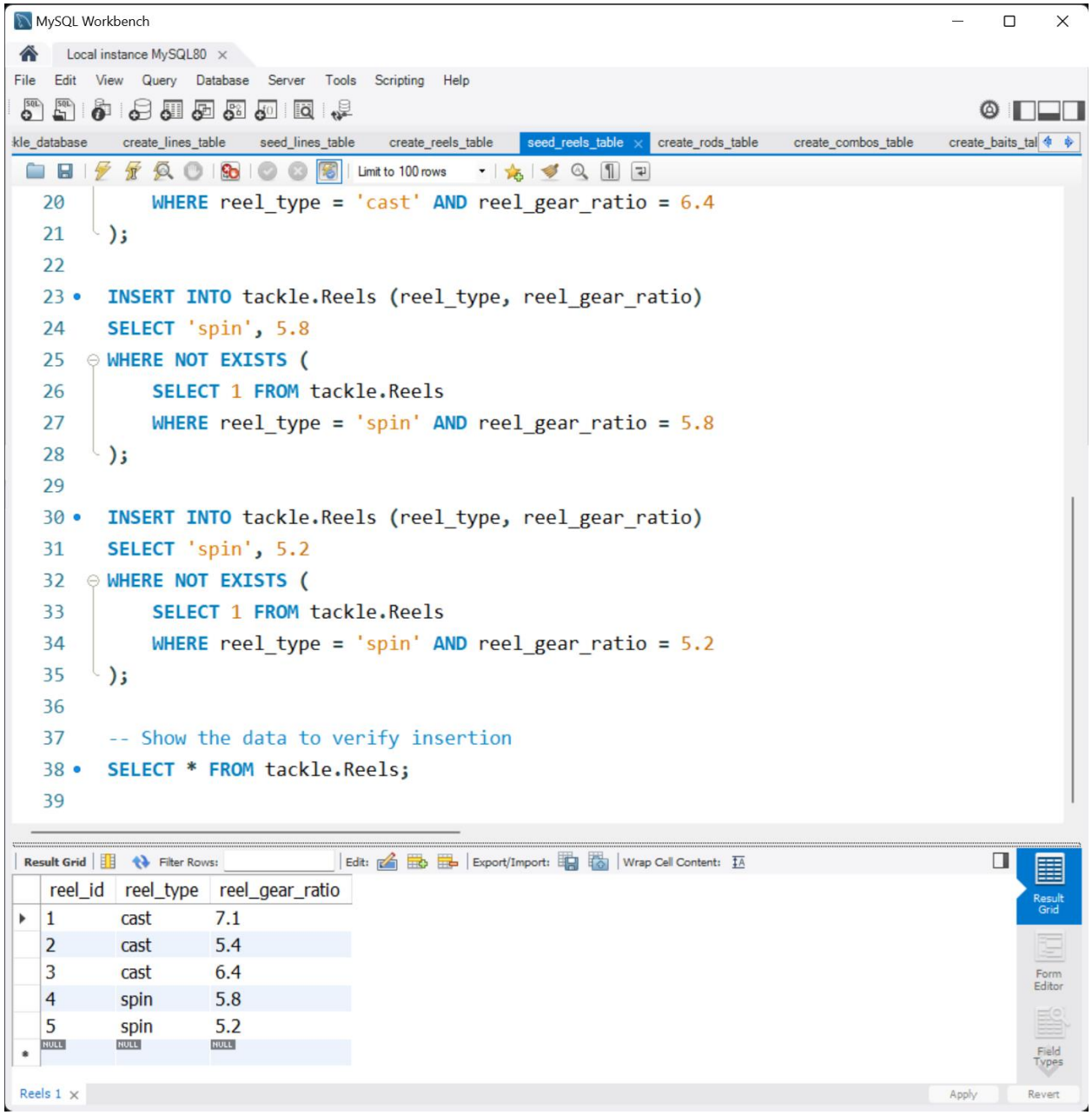
Create Reels Table



Note. Code at
https://github.com/ciioprof0/tackle/blob/main/db/schema/tables/create_reels_table.sql

Figure 8

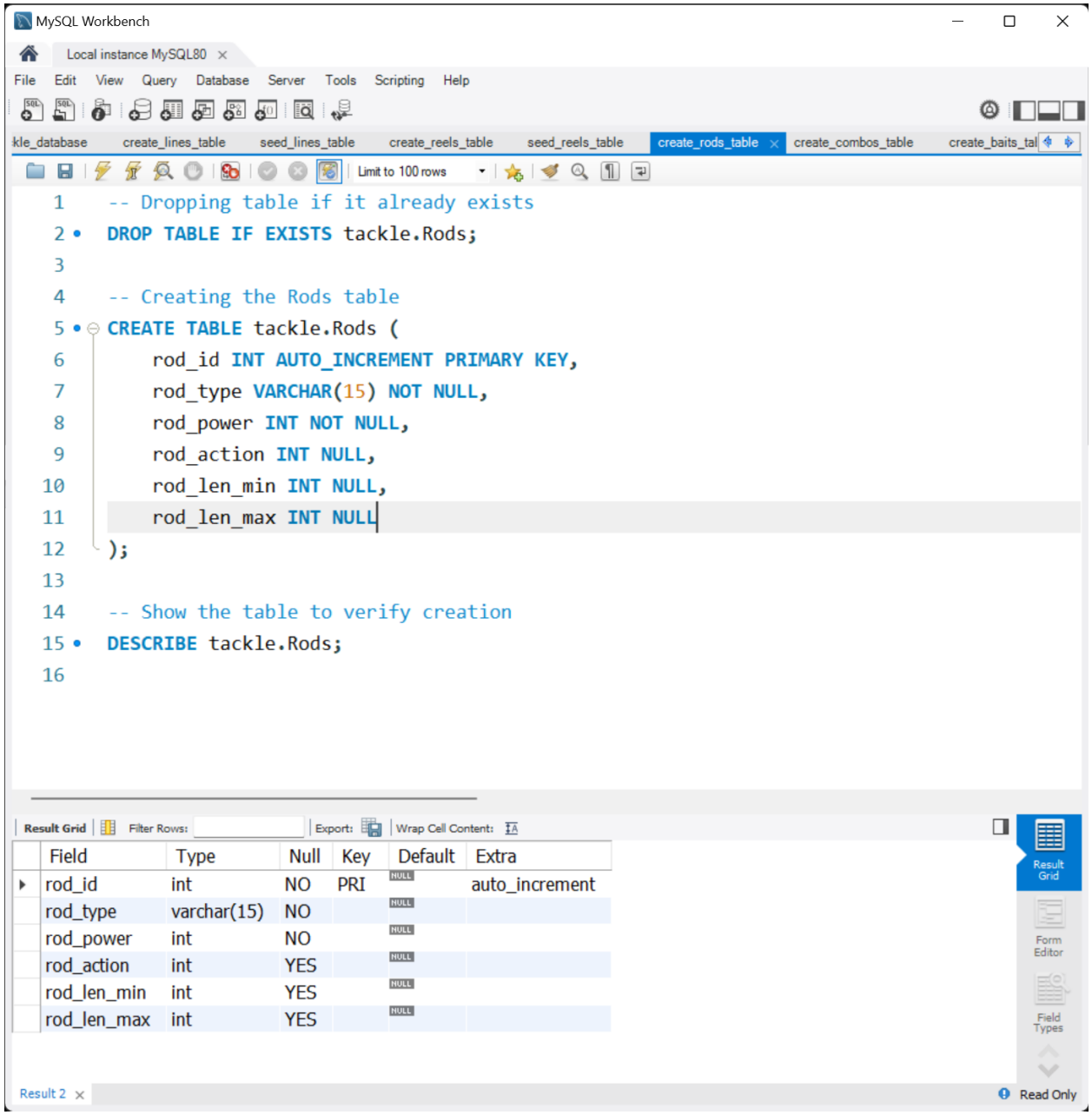
Seed Reels Table



Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/seeds/seed_reels_table.sql
Seed data at https://github.com/ciioprof0/tackle/raw/main/data/raw/data.xlsx

Figure 9

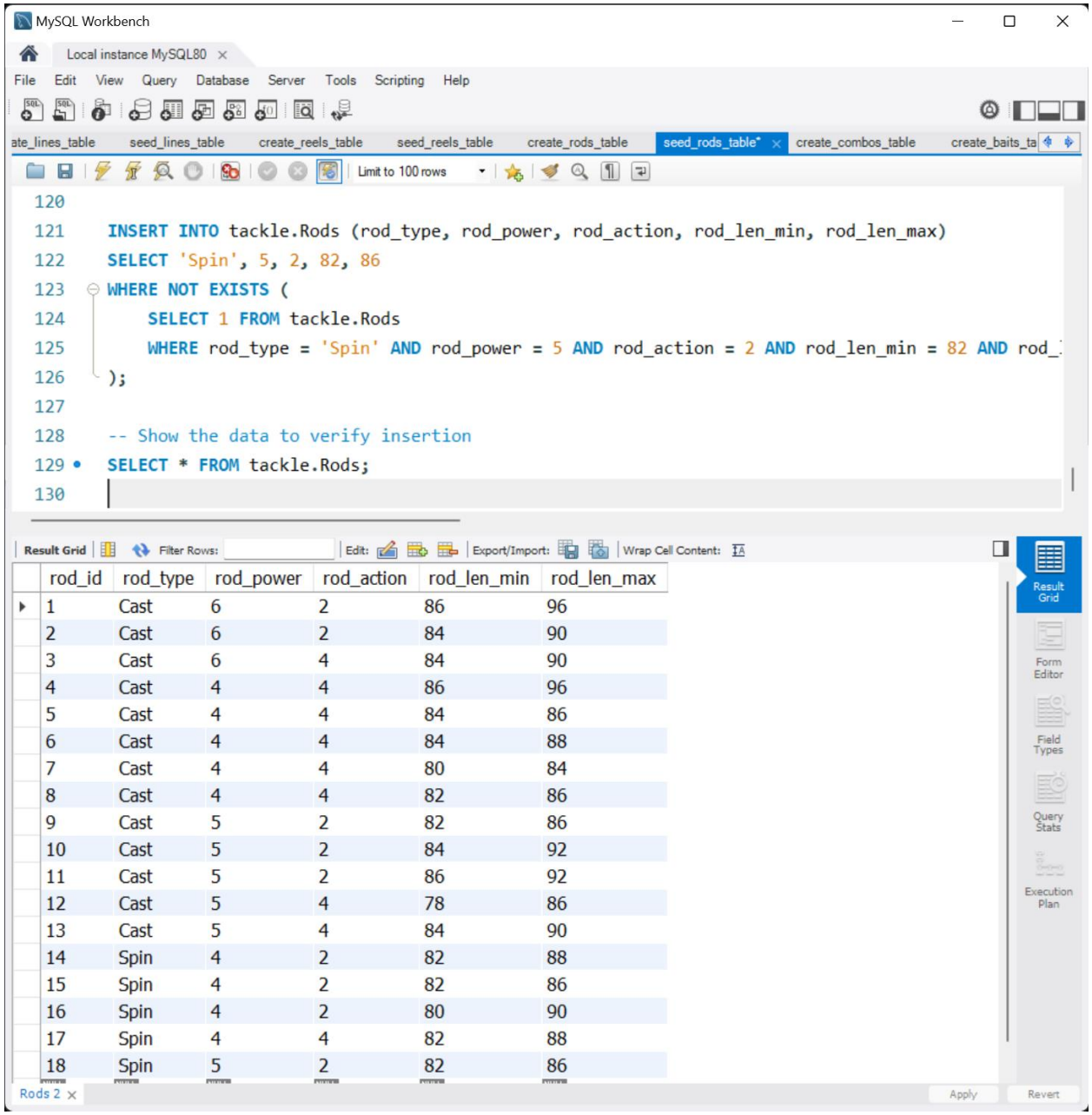
Create Rods Table



Note. Code at
https://github.com/ciioprof0/tackle/blob/main/db/schema/tables/create_rods_table.sql

Figure 10

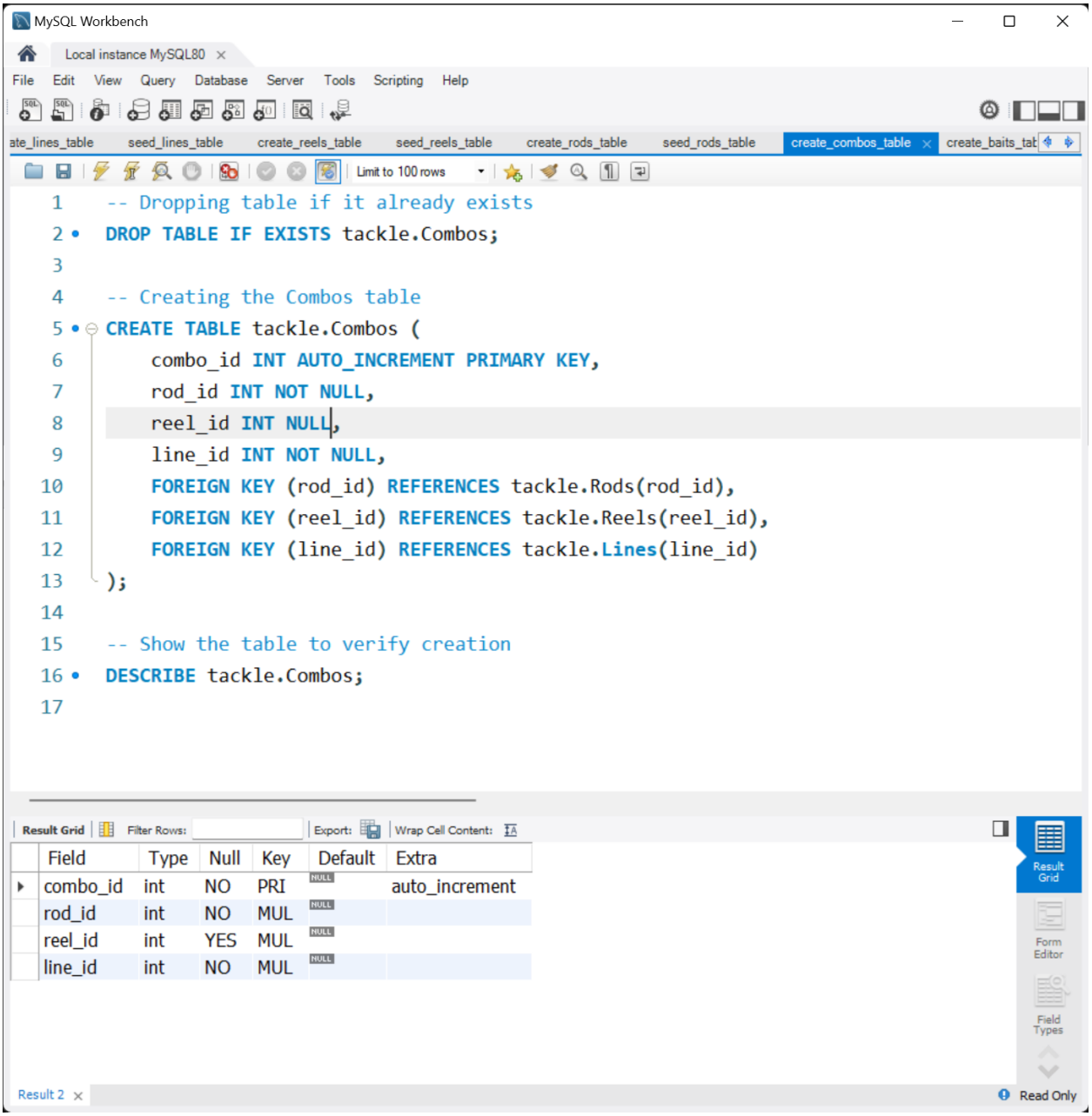
Seed Rods Table



Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/seeds/seed_rods_table.sql
Seed data at https://github.com/ciioprof0/tackle/raw/main/data/raw/data.xlsx

Figure 11

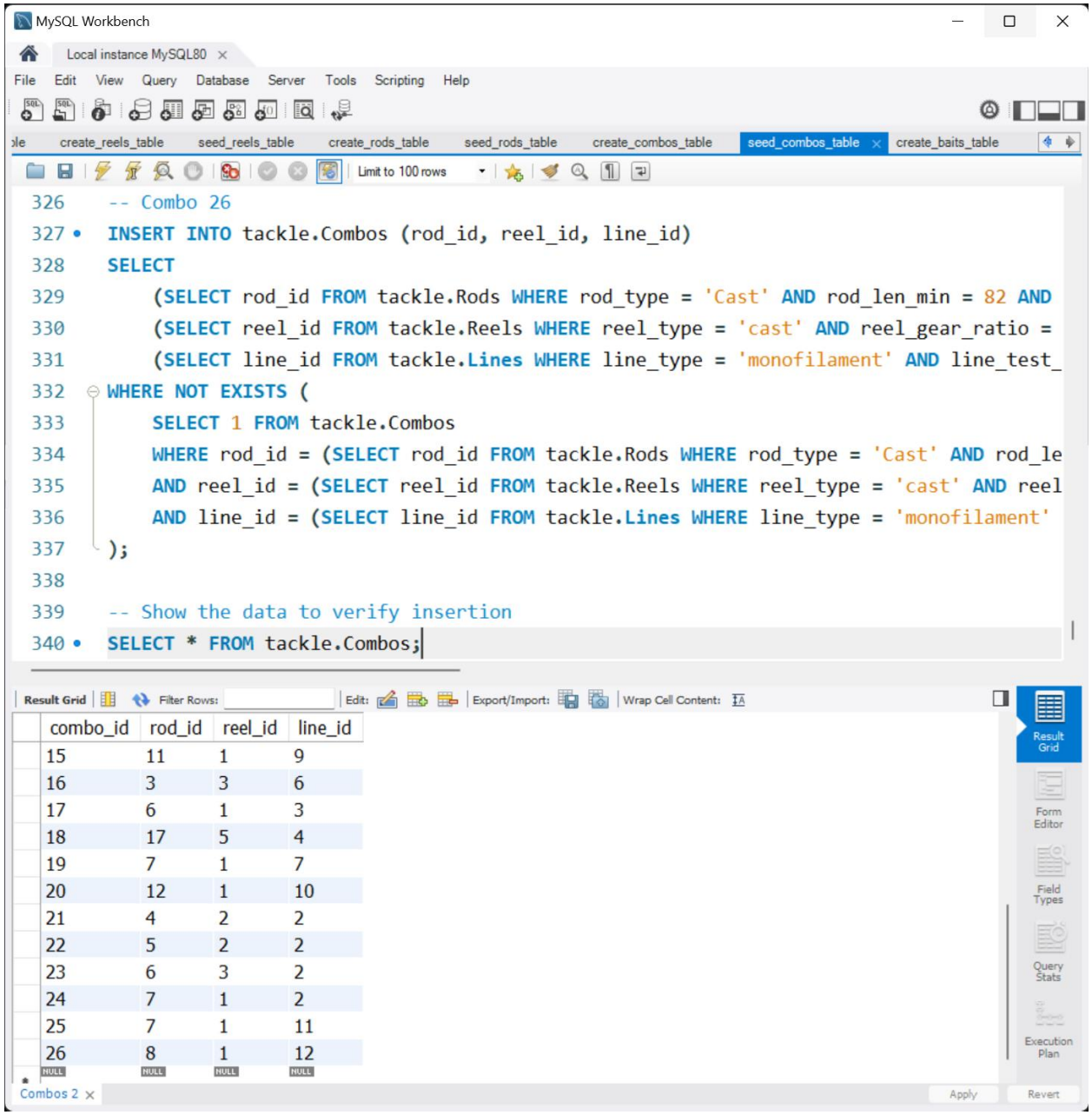
Create Combos Table



Note. Code at
https://github.com/ciioprof0/tackle/blob/main/db/schema/tables/create_combos_table.sql

Figure 12

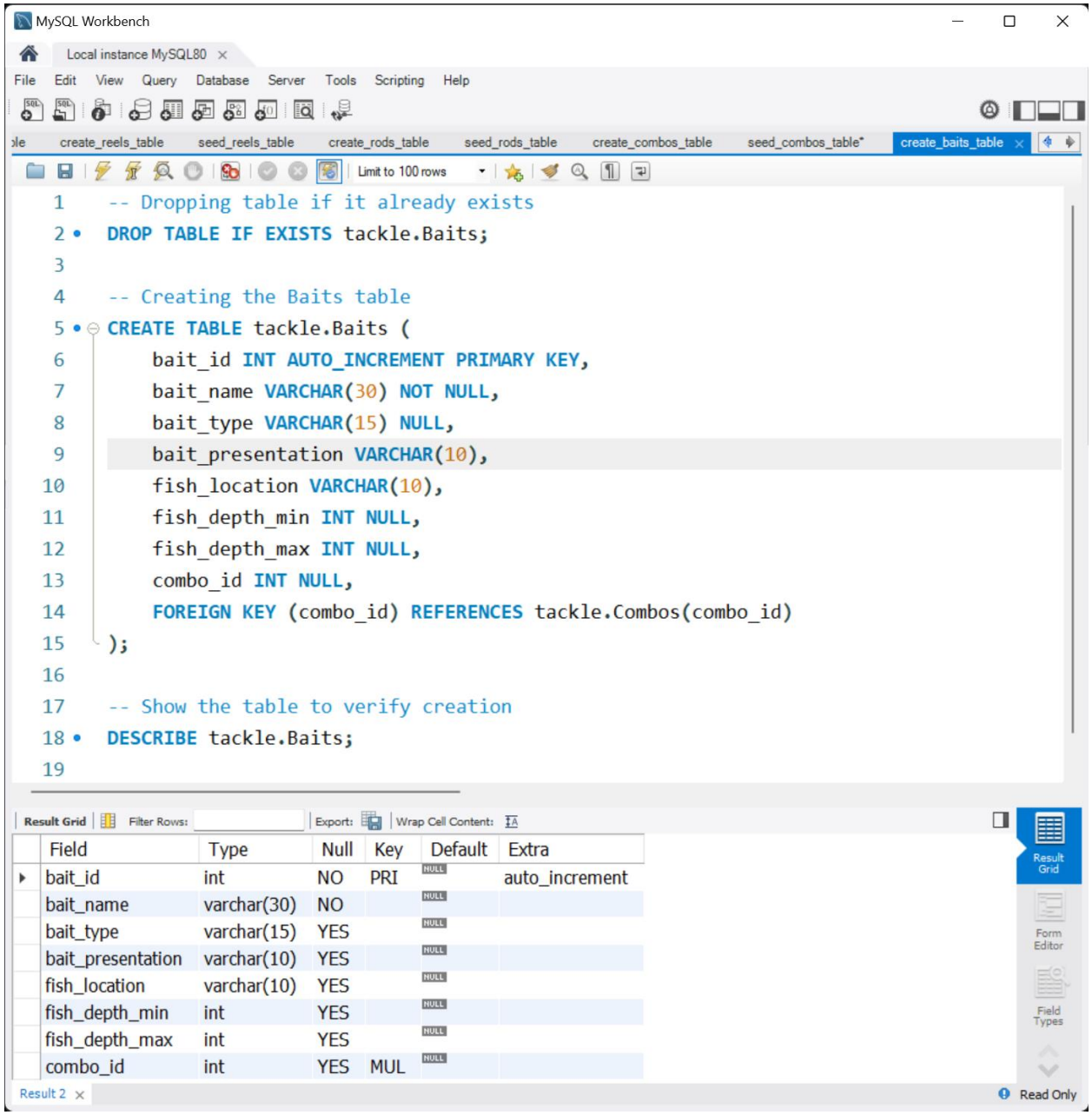
Seed Combos Table



Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/seeds/seed_combos_table.sql
Seed data at https://github.com/ciioprof0/tackle/raw/main/data/raw/data.xlsx

Figure 13

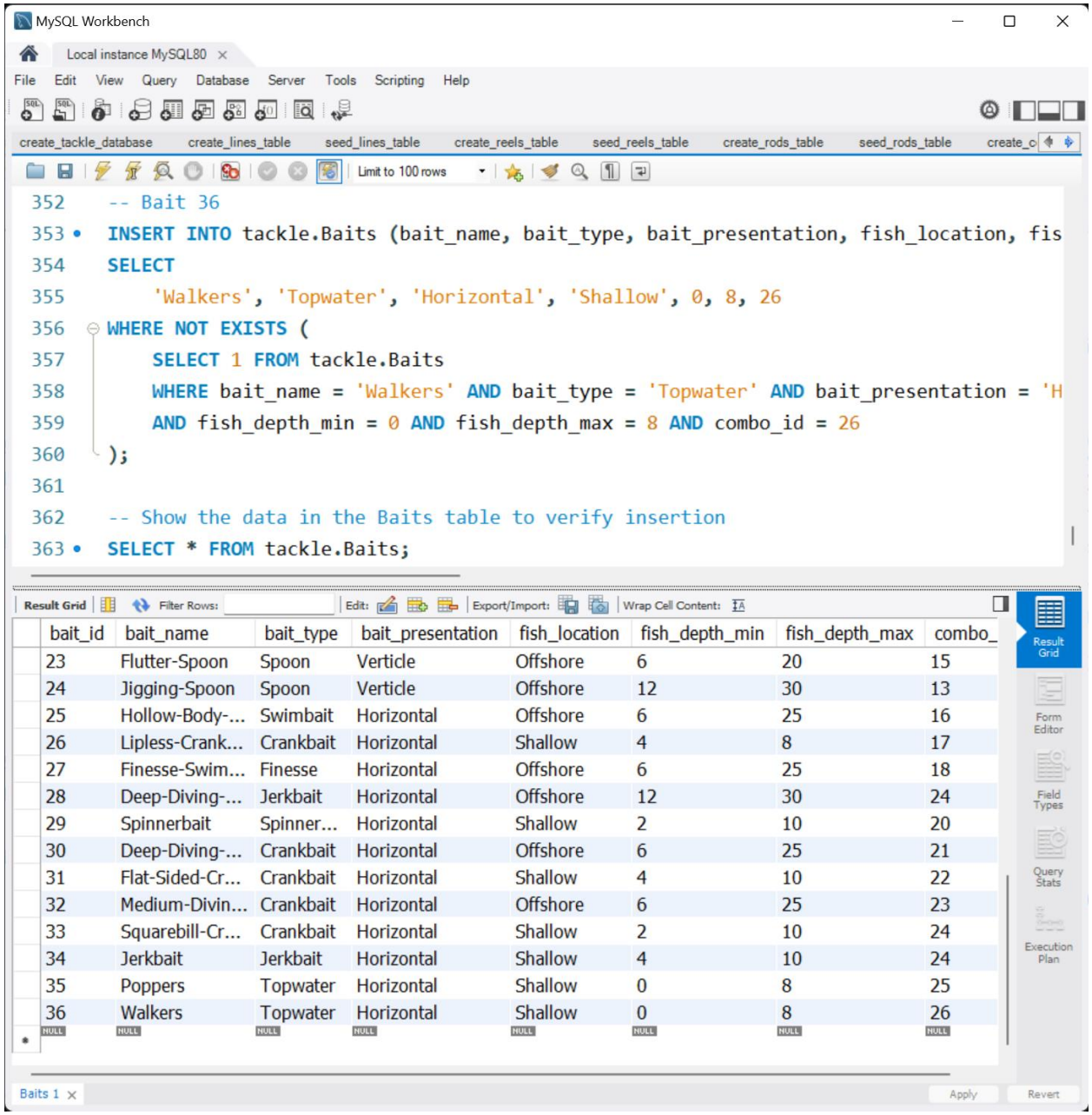
Create Baits Table



Note. Code at
https://github.com/ciioprof0/tackle/blob/main/db/schema/tables/create_baits_table.sql

Figure 14

Seed Baits Table



Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/seeds/seed_baits_table.sql
Seed data at https://github.com/ciioprof0/tackle/raw/main/data/raw/data.xlsx

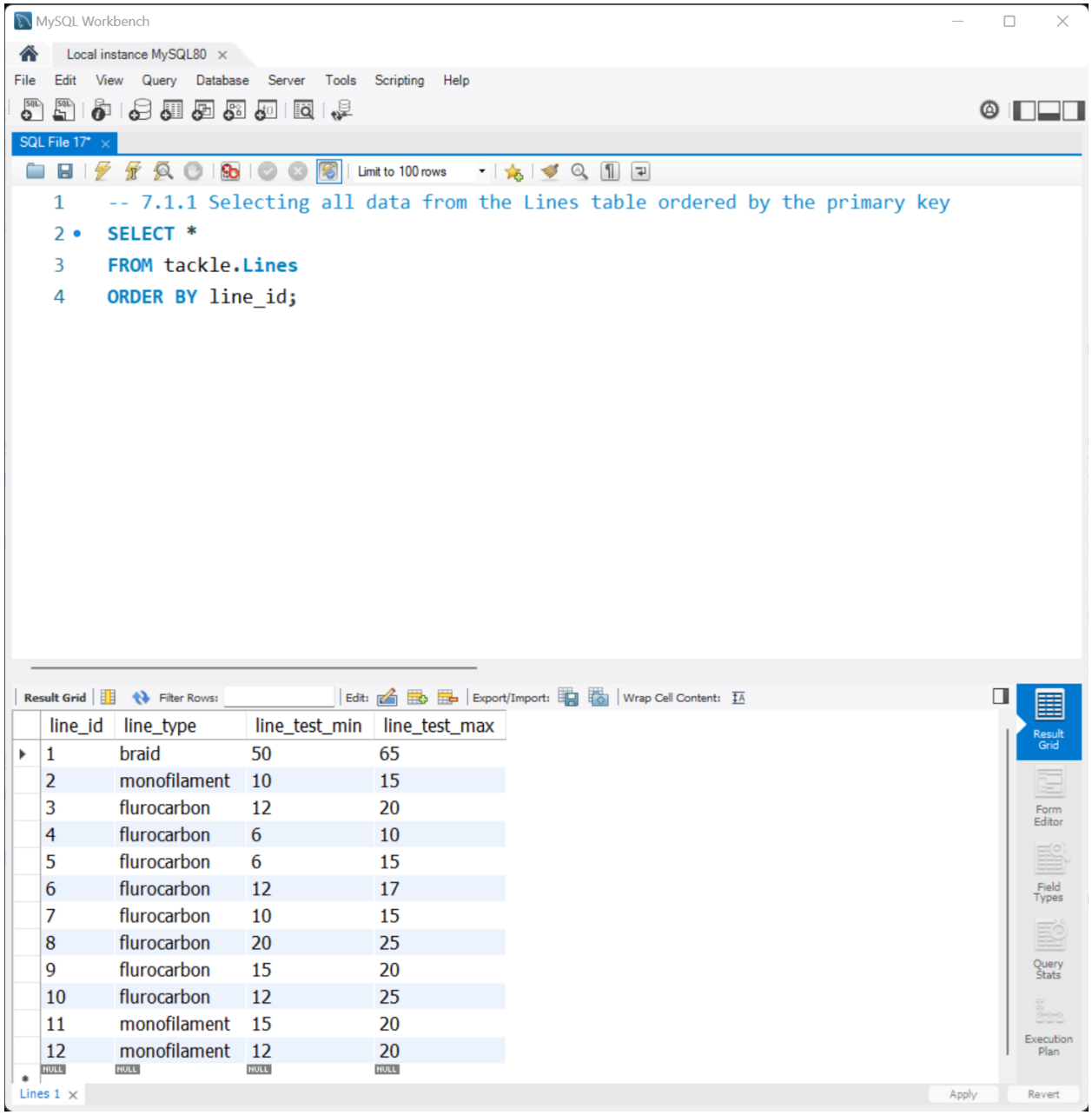
Queries

7. Create a variety of SQL queries to retrieve data from one or many tables:

1. Retrieve the data from each table by using the SELECT * statement and order by PK column(s). Show the output. Make sure you show the print screen of the complete set of rows and columns. The rows must be ordered by PK column(s).

Figure 15

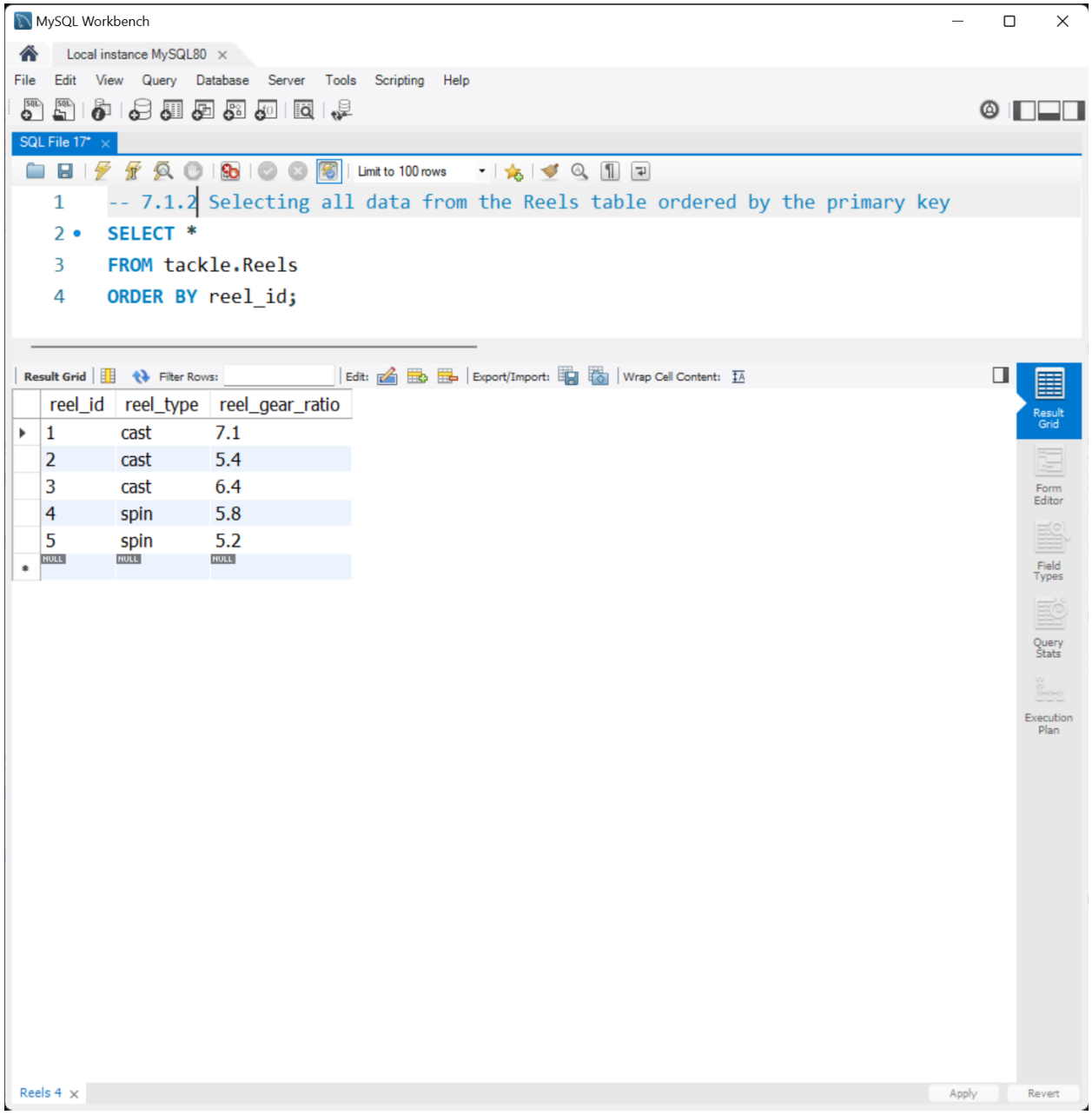
*SELECT * FROM Lines Table*



Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/queries/q71.sql. Data at https://github.com/ciioprof0/tackle/blob/main/data/processed/7_1_1.csv

Figure 16

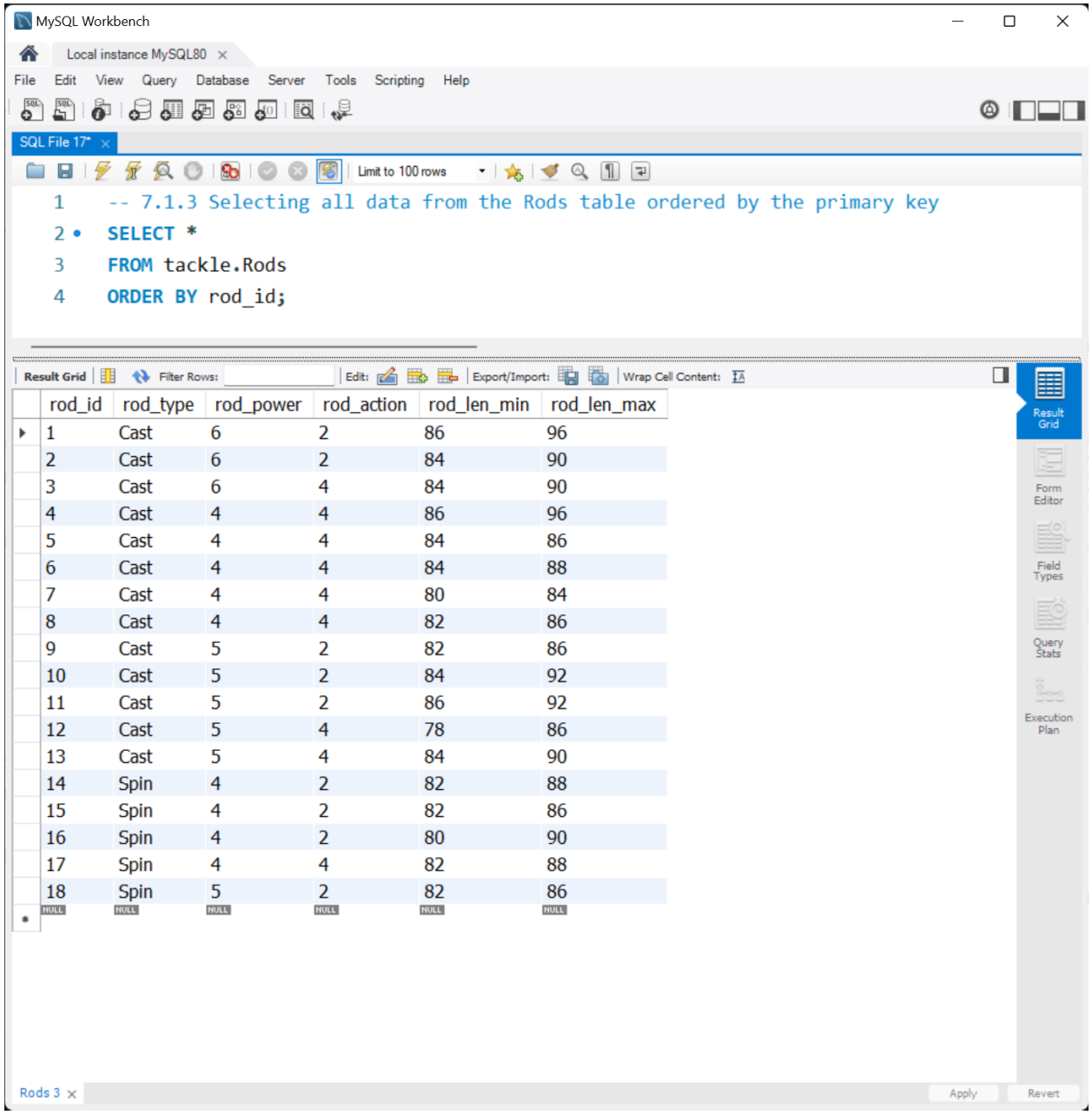
*SELECT * FROM Reels Table*



Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/queries/q71.sql. Data at https://github.com/ciioprof0/tackle/blob/main/data/processed/7_1_2.csv

Figure 17

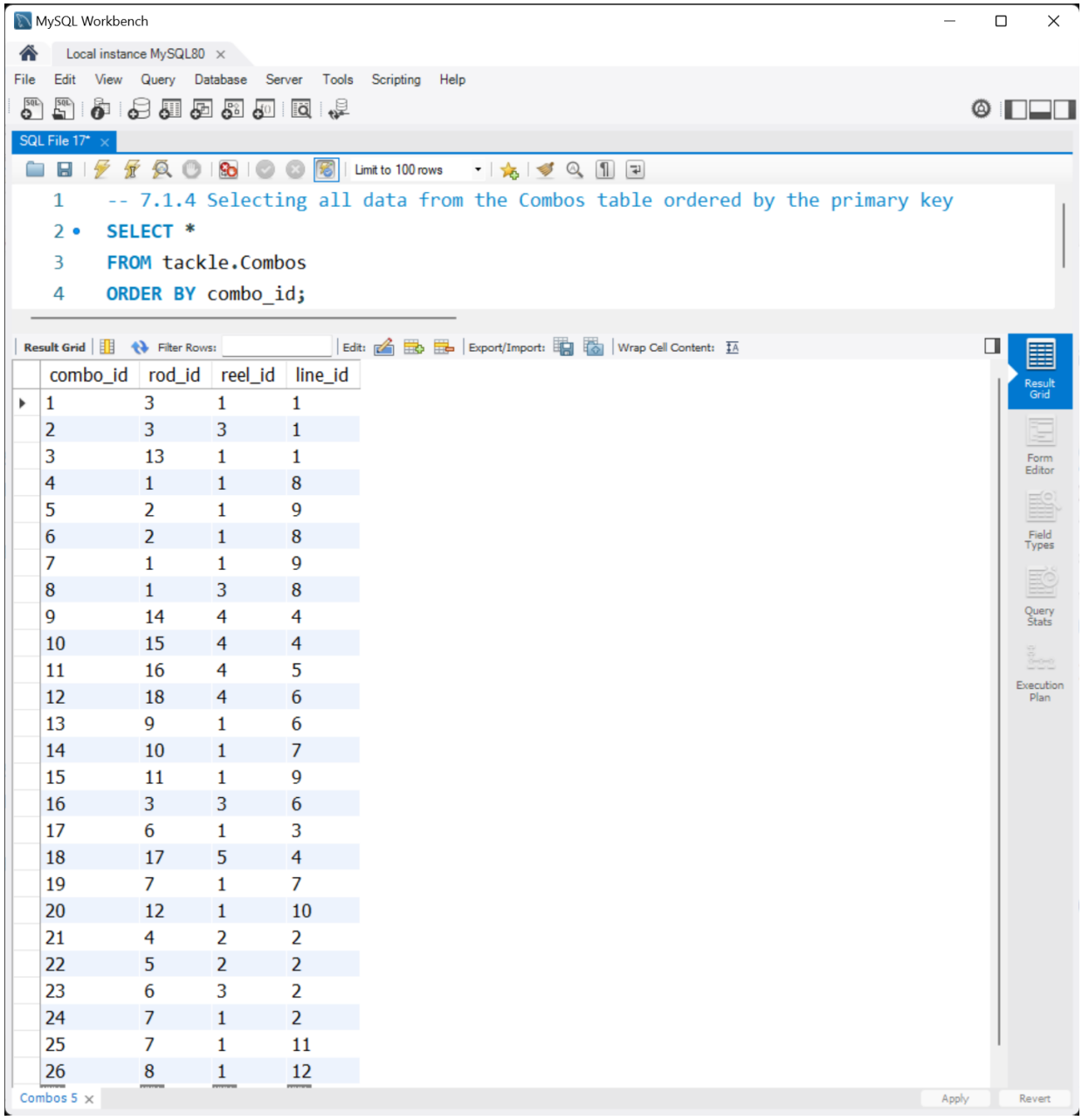
*SELECT * FROM Rods Table*



Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/queries/q71.sql. Data at https://github.com/ciioprof0/tackle/blob/main/data/processed/7_1_3.csv

Figure 18

*SELECT * FROM Combo Table*

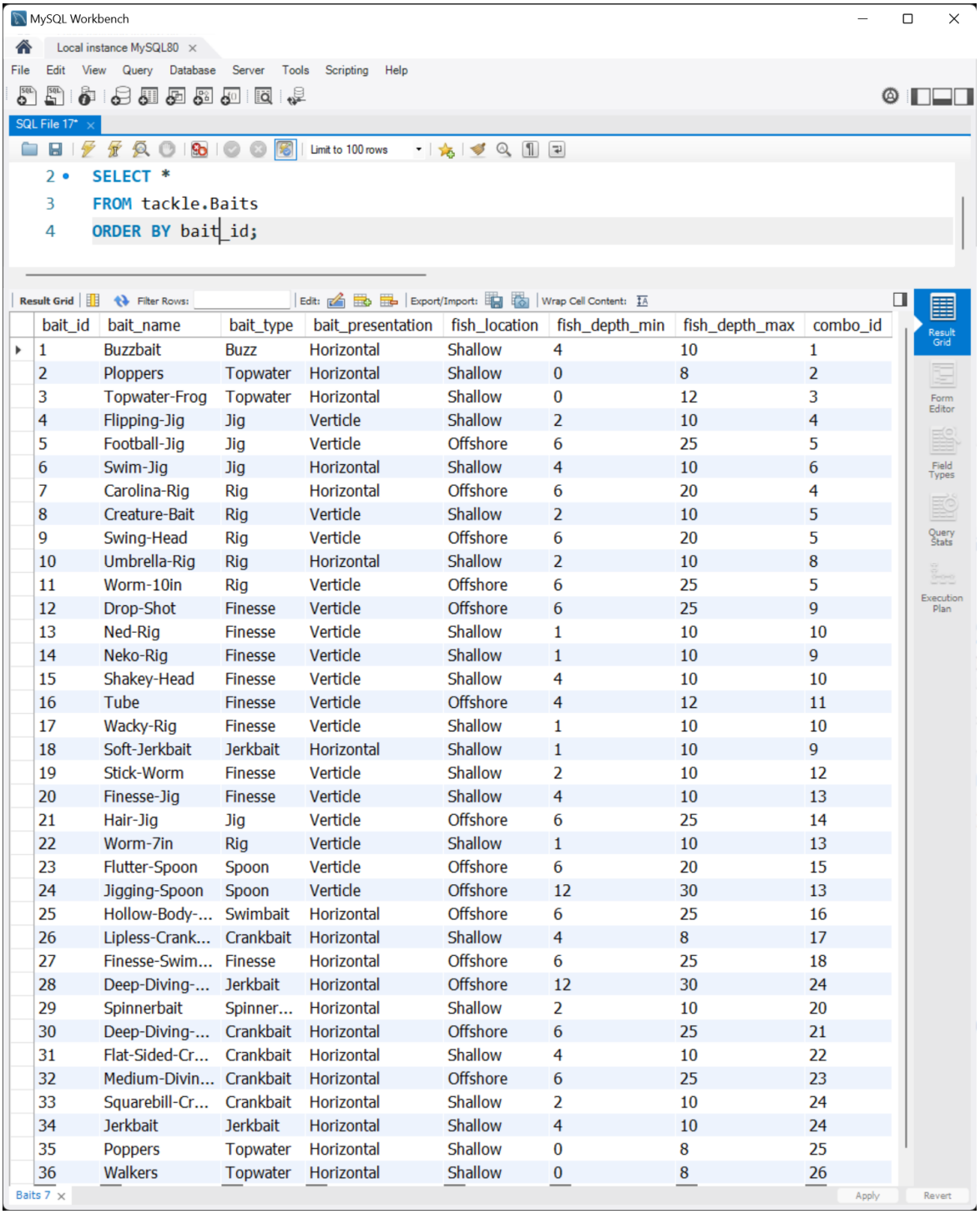


combo_id	rod_id	reel_id	line_id
1	3	1	1
2	3	3	1
3	13	1	1
4	1	1	8
5	2	1	9
6	2	1	8
7	1	1	9
8	1	3	8
9	14	4	4
10	15	4	4
11	16	4	5
12	18	4	6
13	9	1	6
14	10	1	7
15	11	1	9
16	3	3	6
17	6	1	3
18	17	5	4
19	7	1	7
20	12	1	10
21	4	2	2
22	5	2	2
23	6	3	2
24	7	1	2
25	7	1	11
26	8	1	12

Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/queries/q71.sql. Data at https://github.com/ciioprof0/tackle/blob/main/data/processed/7_1_4.csv

Figure 19

*SELECT * FROM Baits Table*



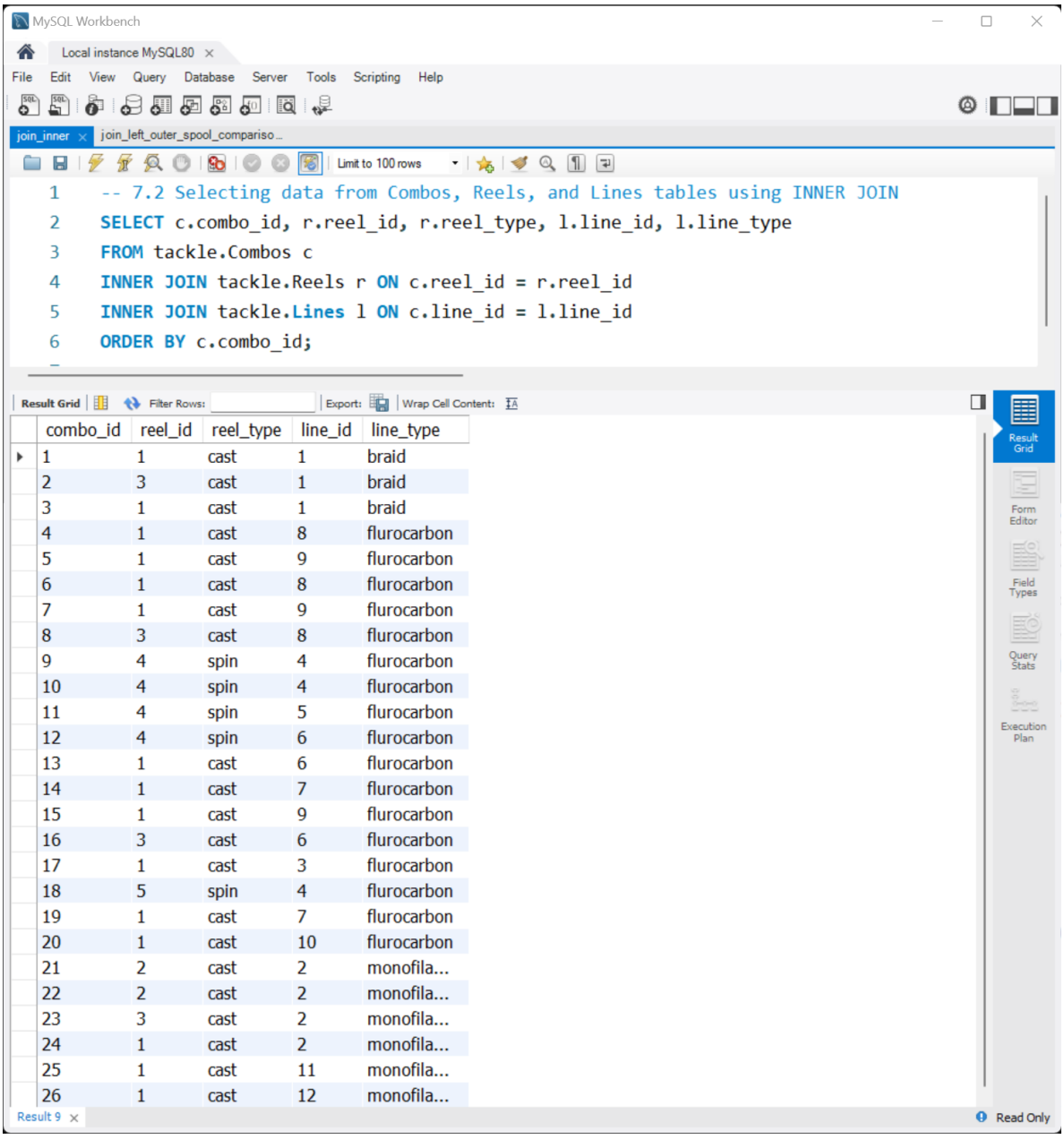
bait_id	bait_name	bait_type	bait_presentation	fish_location	fish_depth_min	fish_depth_max	combo_id
1	Buzzbait	Buzz	Horizontal	Shallow	4	10	1
2	Ploppers	Topwater	Horizontal	Shallow	0	8	2
3	Topwater-Frog	Topwater	Horizontal	Shallow	0	12	3
4	Flipping-Jig	Jig	Verticle	Shallow	2	10	4
5	Football-Jig	Jig	Verticle	Offshore	6	25	5
6	Swim-Jig	Jig	Horizontal	Shallow	4	10	6
7	Carolina-Rig	Rig	Horizontal	Offshore	6	20	4
8	Creature-Bait	Rig	Verticle	Shallow	2	10	5
9	Swing-Head	Rig	Verticle	Offshore	6	20	5
10	Umbrella-Rig	Rig	Horizontal	Shallow	2	10	8
11	Worm-10in	Rig	Verticle	Offshore	6	25	5
12	Drop-Shot	Finesse	Verticle	Offshore	6	25	9
13	Ned-Rig	Finesse	Verticle	Shallow	1	10	10
14	Neko-Rig	Finesse	Verticle	Shallow	1	10	9
15	Shakey-Head	Finesse	Verticle	Shallow	4	10	10
16	Tube	Finesse	Verticle	Offshore	4	12	11
17	Wacky-Rig	Finesse	Verticle	Shallow	1	10	10
18	Soft-Jerkbait	Jerkbait	Horizontal	Shallow	1	10	9
19	Stick-Worm	Finesse	Verticle	Shallow	2	10	12
20	Finesse-Jig	Finesse	Verticle	Shallow	4	10	13
21	Hair-Jig	Jig	Verticle	Offshore	6	25	14
22	Worm-7in	Rig	Verticle	Shallow	1	10	13
23	Flutter-Spoon	Spoon	Verticle	Offshore	6	20	15
24	Jigging-Spoon	Spoon	Verticle	Offshore	12	30	13
25	Hollow-Body-...	Swimbait	Horizontal	Offshore	6	25	16
26	Lipless-Crank...	Crankbait	Horizontal	Shallow	4	8	17
27	Finesse-Swim...	Finesse	Horizontal	Offshore	6	25	18
28	Deep-Diving-...	Jerkbait	Horizontal	Offshore	12	30	24
29	Spinnerbait	Spinner...	Horizontal	Shallow	2	10	20
30	Deep-Diving-...	Crankbait	Horizontal	Offshore	6	25	21
31	Flat-Sided-Cr...	Crankbait	Horizontal	Shallow	4	10	22
32	Medium-Divin...	Crankbait	Horizontal	Offshore	6	25	23
33	Squarebill-Cr...	Crankbait	Horizontal	Shallow	2	10	24
34	Jerkbait	Jerkbait	Horizontal	Shallow	4	10	24
35	Poppers	Topwater	Horizontal	Shallow	0	8	25
36	Walkers	Topwater	Horizontal	Shallow	0	8	26

Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/queries/q71.sql. Data at https://github.com/ciioprof0/tackle/blob/main/data/processed/7_1_5.csv

2. Write an SQL involving the junction table and two other related tables. You must use the INNER JOIN to connect with all three tables. The database that you created must be included in your SQL queries.

Figure 20

INNER JOIN Connecting Three Tables



combo_id	reel_id	reel_type	line_id	line_type
1	1	cast	1	braid
2	3	cast	1	braid
3	1	cast	1	braid
4	1	cast	8	flurocarbon
5	1	cast	9	flurocarbon
6	1	cast	8	flurocarbon
7	1	cast	9	flurocarbon
8	3	cast	8	flurocarbon
9	4	spin	4	flurocarbon
10	4	spin	4	flurocarbon
11	4	spin	5	flurocarbon
12	4	spin	6	flurocarbon
13	1	cast	6	flurocarbon
14	1	cast	7	flurocarbon
15	1	cast	9	flurocarbon
16	3	cast	6	flurocarbon
17	1	cast	3	flurocarbon
18	5	spin	4	flurocarbon
19	1	cast	7	flurocarbon
20	1	cast	10	flurocarbon
21	2	cast	2	monofila...
22	2	cast	2	monofila...
23	3	cast	2	monofila...
24	1	cast	2	monofila...
25	1	cast	11	monofila...
26	1	cast	12	monofila...

Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/queries/q72.sql. Data at https://github.com/ciioprof0/tackle/blob/main/data/processed/7_2.csv

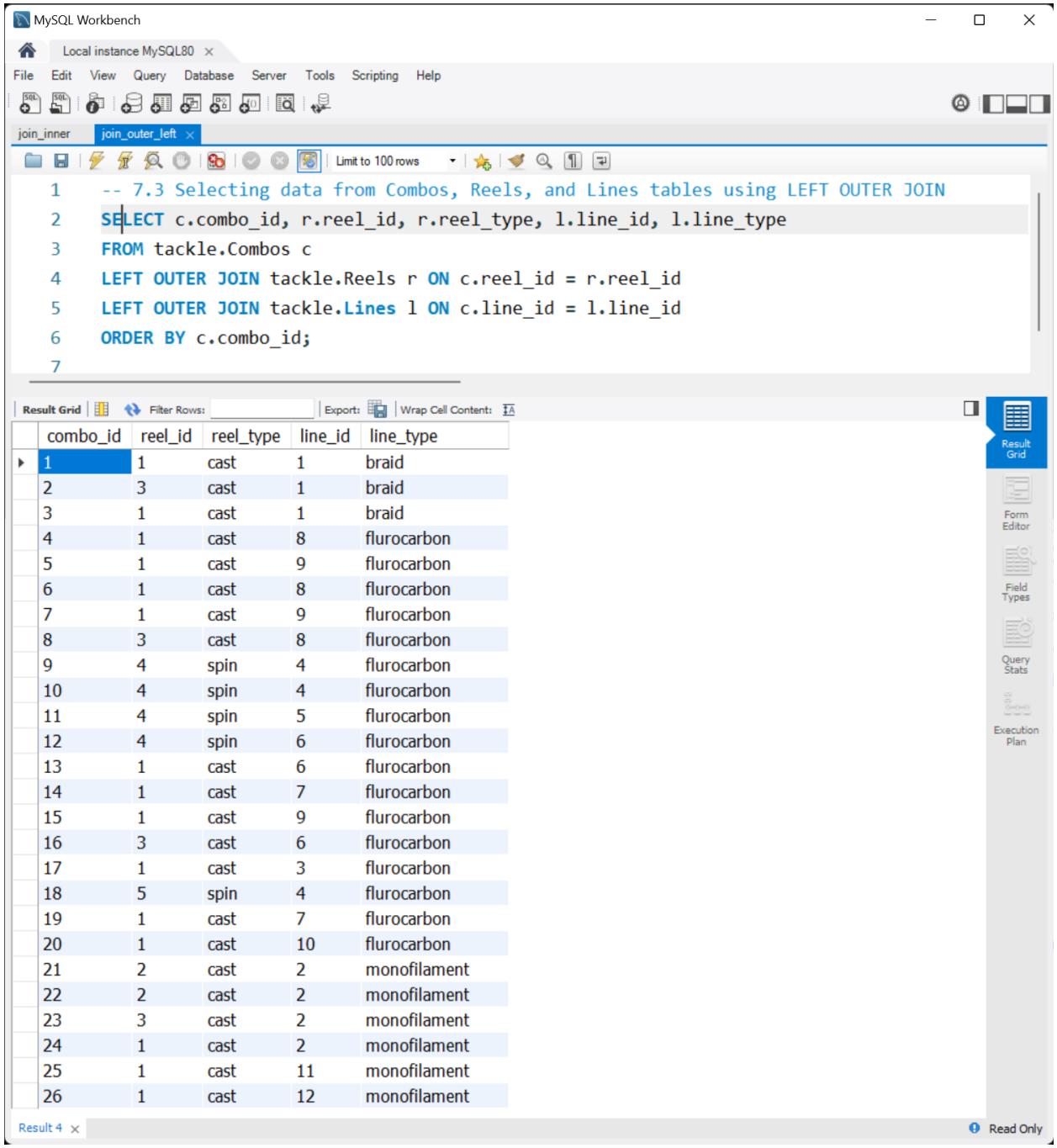
3. Write an SQL by including two or more tables and using the LEFT OUTER JOIN. Show the results and sort the results by key field(s). Interpret the results compared to what an INNER JOIN does.

The results are the same because the `tackle` dataset is "complete" regarding relationships between the Combos, Reels, and Lines tables. Since every `combo_id` has matching `reel_id` and `line_id`, both types of joins return identical results.

To demonstrate the difference between INNER JOIN and LEFT OUTER JOIN, we could introduce a scenario where a `combo_id` references a `reel_id` or `line_id` that does not exist in the corresponding tables. This would allow us to see how LEFT OUTER JOIN behaves differently by including rows with NULL values.

Figure 21

LEFT OUTER JOIN Connecting Three Tables



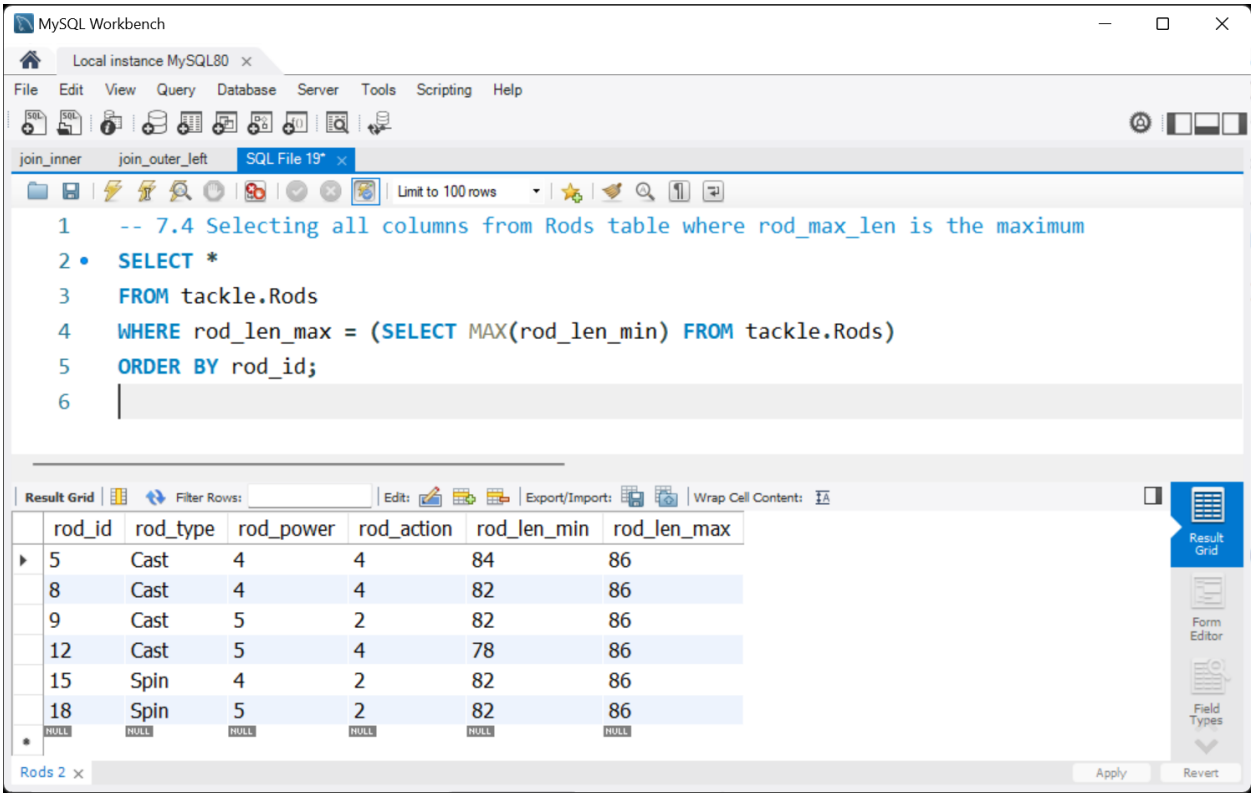
Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/queries/q73.sql. Data at https://github.com/ciioprof0/tackle/blob/main/data/processed/7_3.csv

4. Write a single-row subquery. Show the results and sort the results by key field(s). Interpret the output.

The results show there are six rods whose maximum length (`rod_len_max`) equals the longest minimum length (`rod_len_min`) rod(s). The variability in `rod_len_min` suggests differences in the rods' minimum usable lengths, but all reach the same maximum extension, highlighting their potential for extended use.

Figure 22

Single-Row Subquery



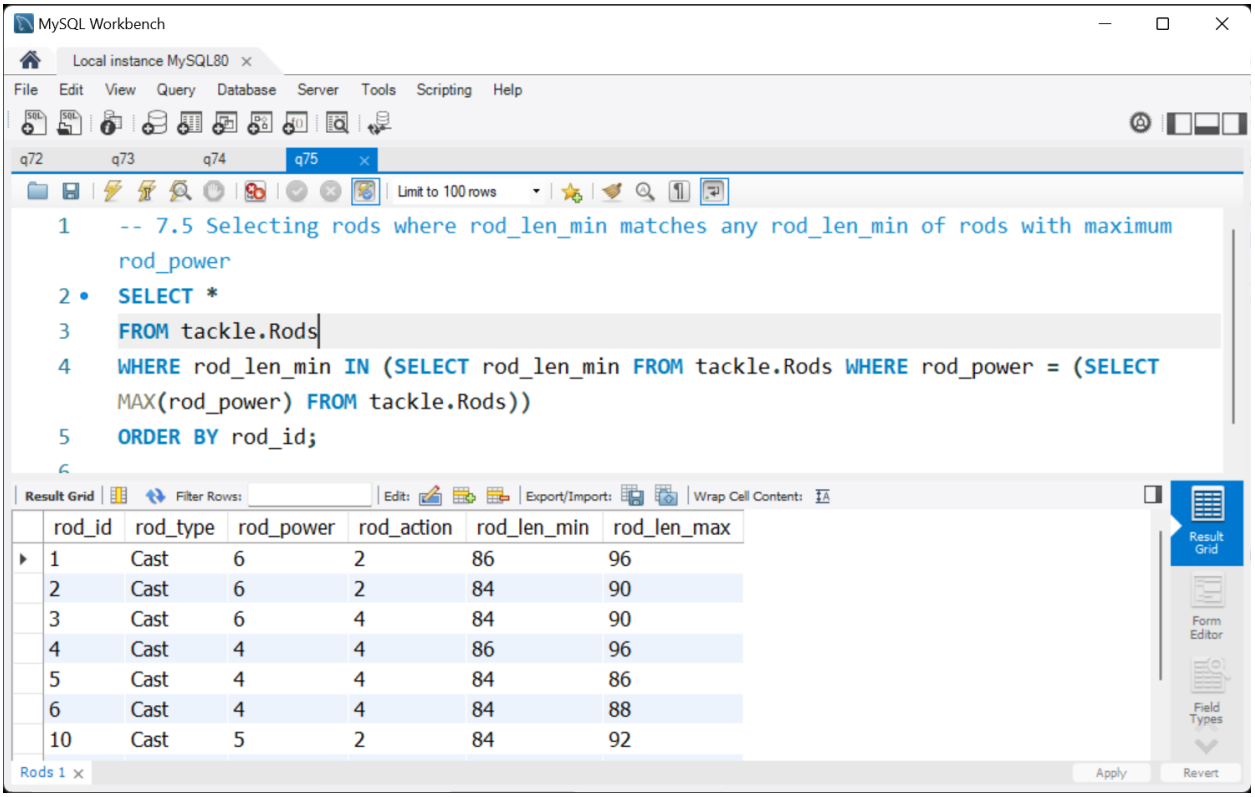
Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/queries/q74.sql. Data at https://github.com/ciioprof0/tackle/blob/main/data/processed/7_4.csv

5. Write a multiple-row subquery. Show the results and sort the results by key field(s). Interpret the output.

The results show rods with a `rod_len_min` matching the minimum lengths of rods with the highest power (`rod_power = 6`). The query highlights that rods with varying power levels (`4`, `5`, and `6`) share similar minimum lengths (`84` or `86`), indicating a common design choice in rod lengths across different power categories. All rods listed are of the "Cast" type, suggesting uniformity in design within this category.

Figure 23

Multiple-Row Subquery



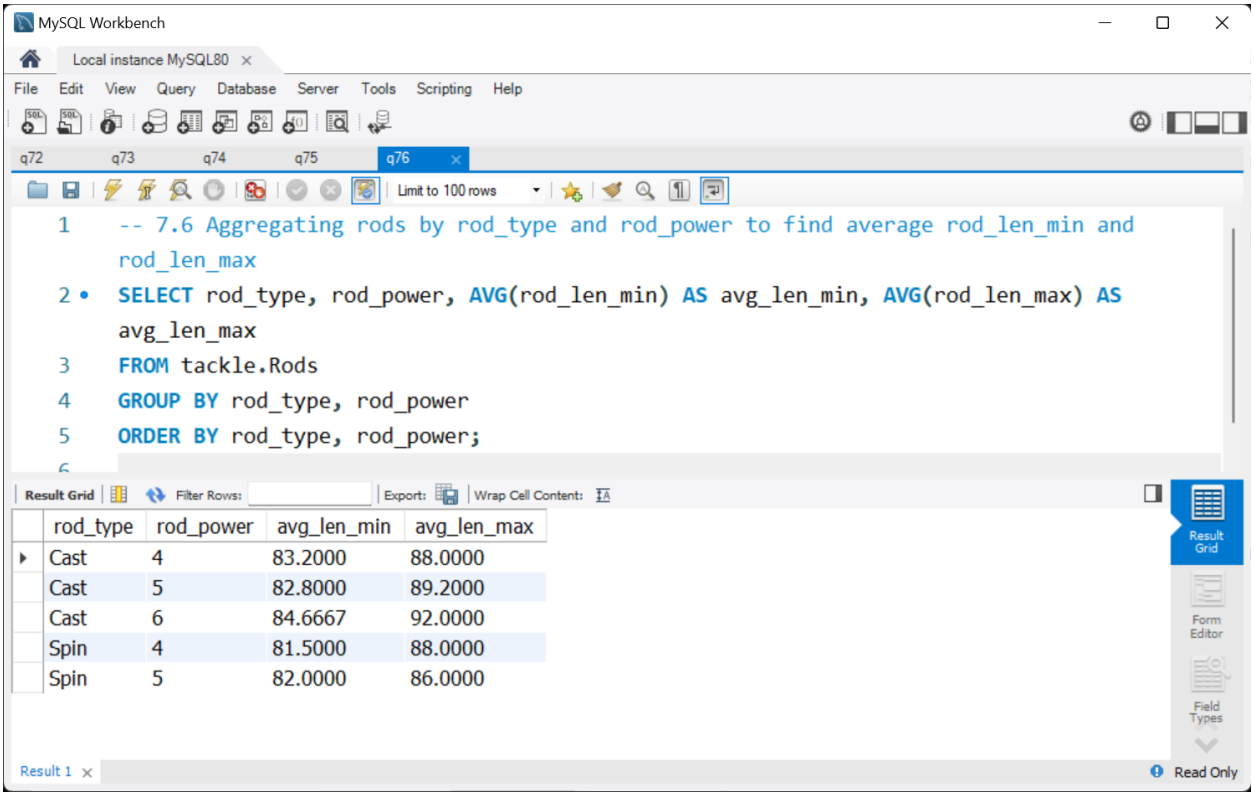
Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/queries/q75.sql. Data at https://github.com/ciioprof0/tackle/blob/main/data/processed/7_5.csv

6. Write an SQL to aggregate the results by using multiple columns in the SELECT clause. Interpret the output.

The output shows how rods' average minimum and maximum lengths vary based on their type (rod_type) and power (rod_power). This information can help identify trends, such as whether certain rod types or power levels tend to have longer or shorter rods on average, offering insights into the design patterns or usage of different rod categories.

Figure 24

Aggregation Using Multiple Columns



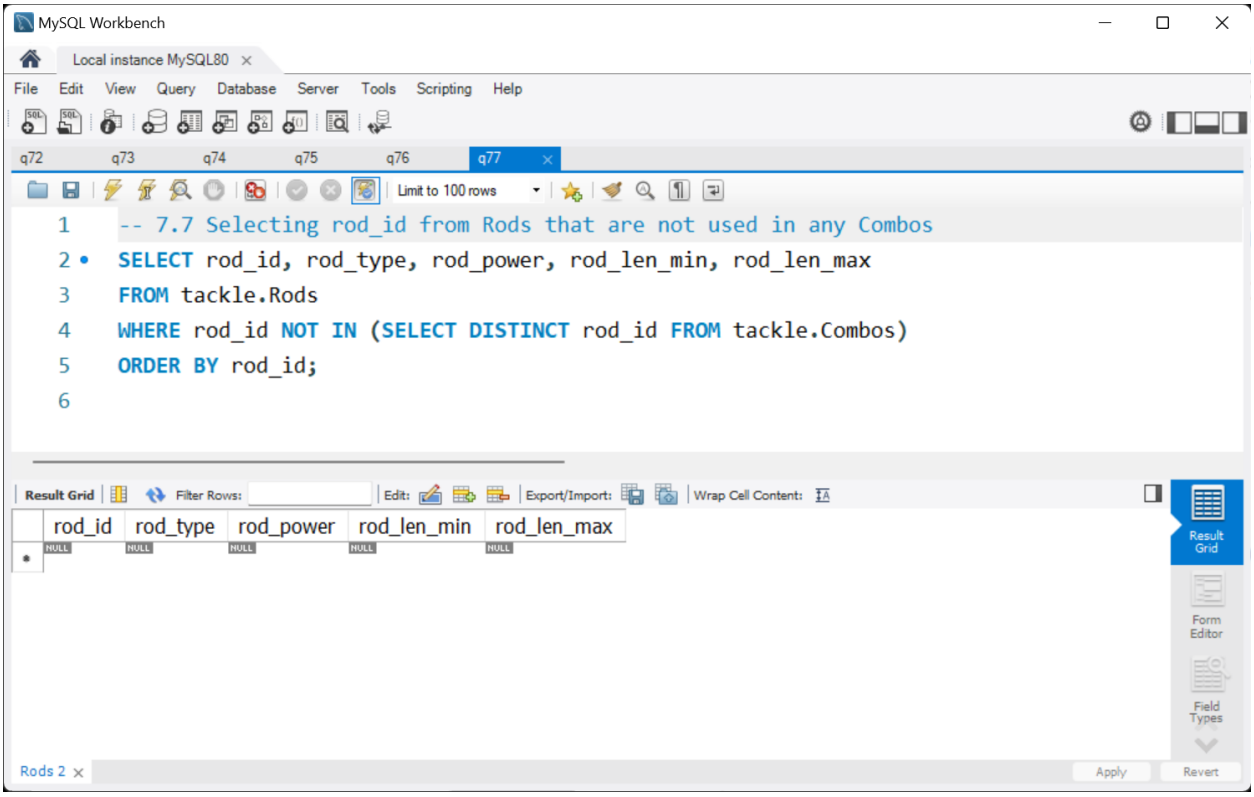
Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/queries/q76.sql. Data at https://github.com/ciioprof0/tackle/blob/main/data/processed/7_6.csv

7. Write a subquery using the NOT IN operator. Show the results and sort the results by key field(s). Interpret the output.

The query will return a list of rods not associated with combinations in the Combos table. This could indicate available rods that have not been paired with a reel and line in the dataset. Since there are no results as expected, all rods are currently paired in combinations, indicating a complete dataset with no unused rods.

Figure 25

Subquery Using NOT IN



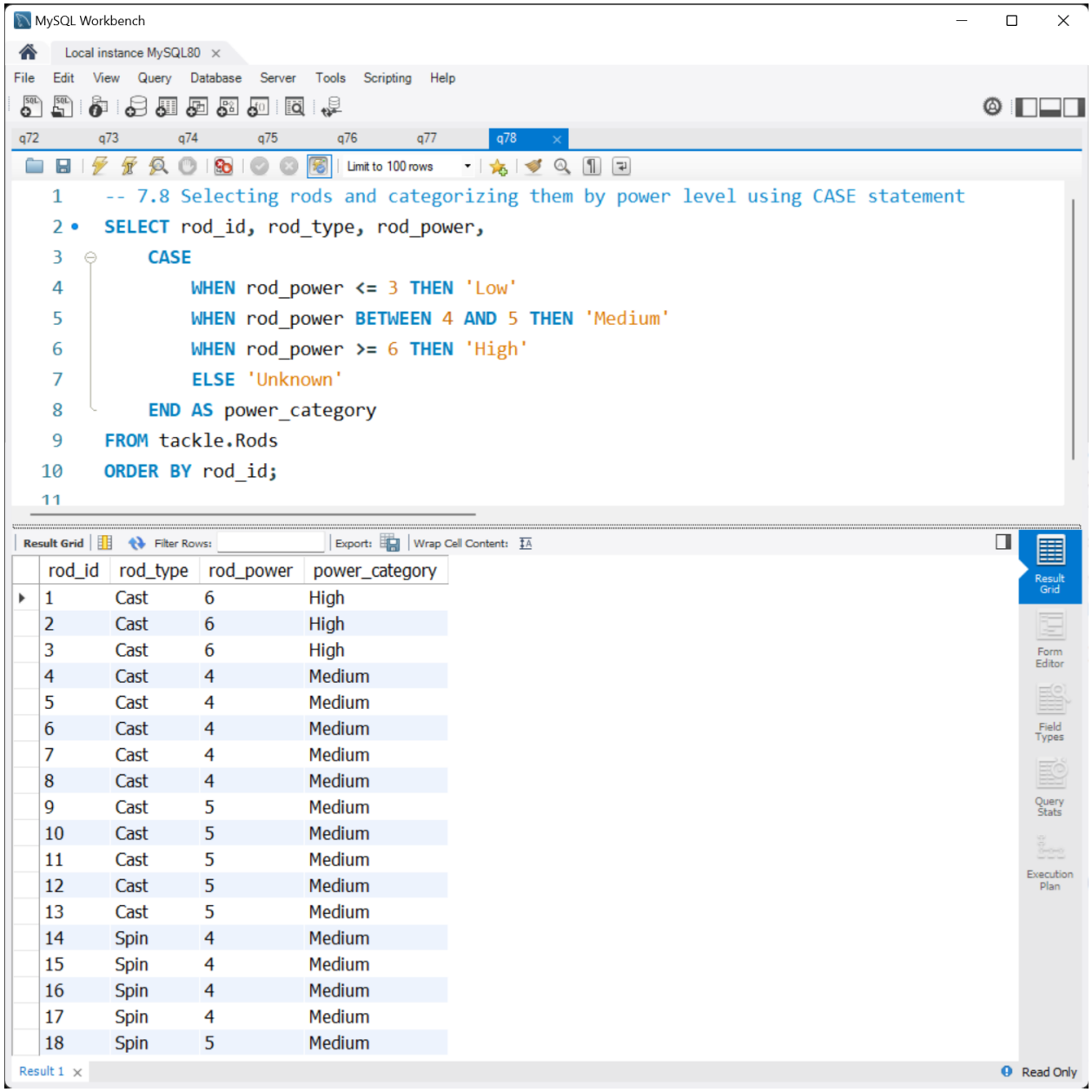
Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/queries/q77.sql.

8. Write a query using a CASE statement. Show the results and sort the results by key field(s). Interpret the output.

The results show that most rods fall into the "Medium" power category, with only a few rods classified as "High" power. All "High" power rods are of the "Cast" type with a rod_power of 6, while the "Medium" power rods include both "Cast" and "Spin" types, with rod_power values ranging from 4 to 5. This distribution indicates that most of the rods in the dataset are designed with medium power, providing a balance of strength suitable for a wide range of fishing conditions. In contrast, fewer rods are made for higher-power applications.

Figure 26

Query Using CASE Statement



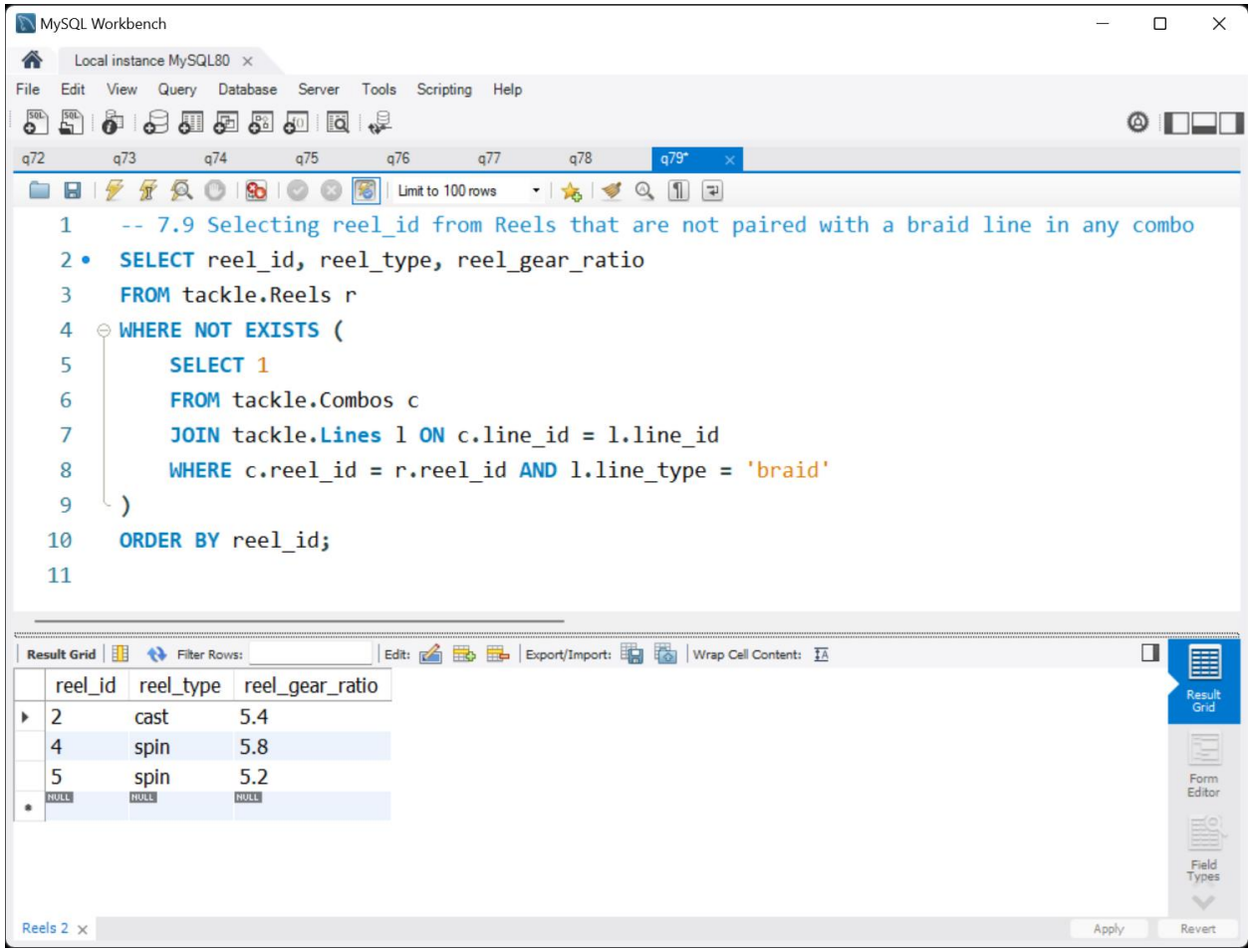
Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/queries/q78.sql. Data at https://github.com/ciioprof0/tackle/blob/main/data/processed/7_8.csv

9. Write a query using the NOT EXISTS operator. Show the results and sort the results by key field(s). Interpret the output.

The results show three reels (`reel_id`s 2, 4, and 5) that have not been paired with a "braid" line in any combo. This suggests that despite being available, these reels have only been used with other types of lines or not at all in the case of "braid." The absence of "braid" line usage with these reels might indicate a preference for other line types with these specific reels or a potential opportunity to explore new combinations with "braid" lines to expand their usage.

Figure 27

Query Using NOT EXISTS Operator



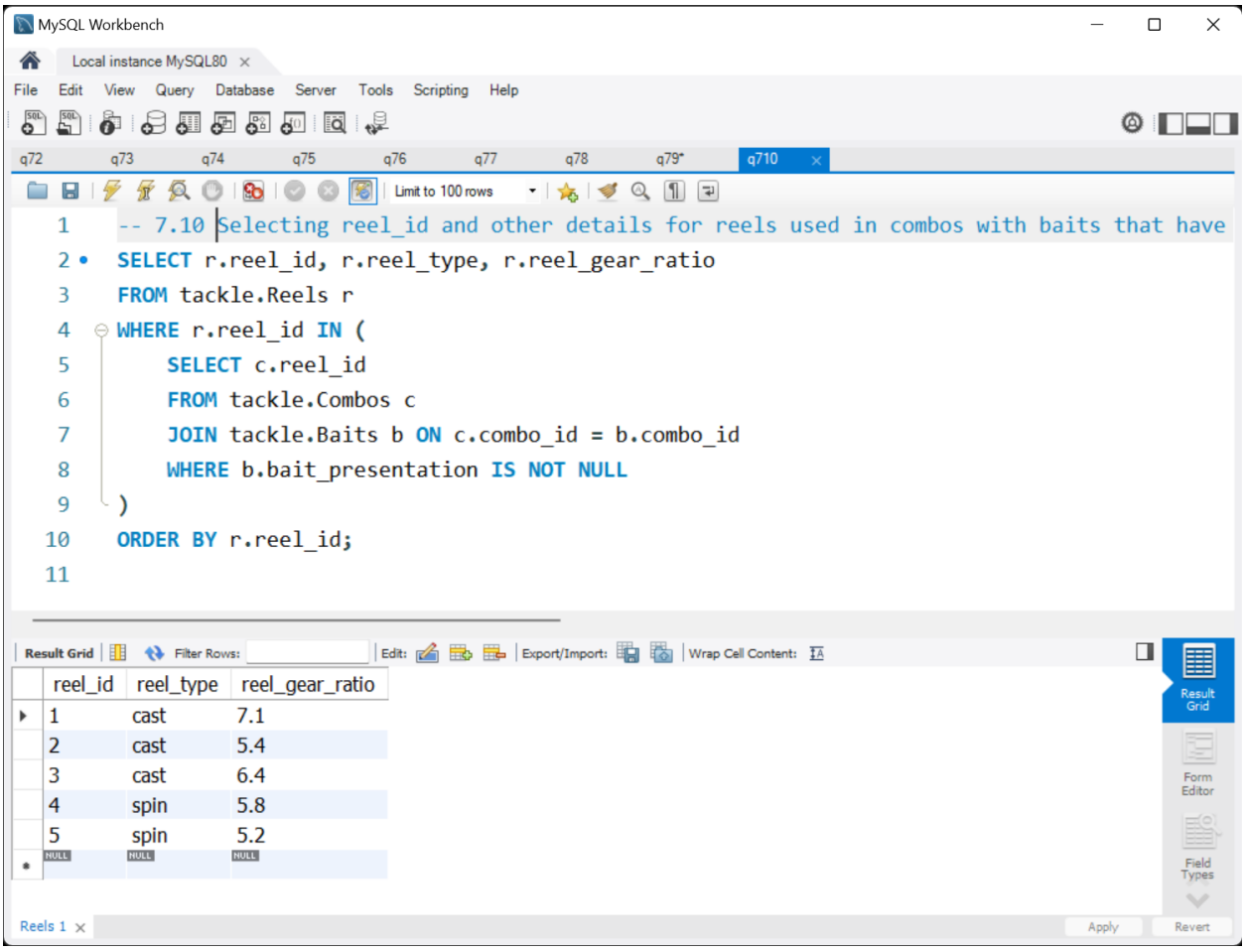
Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/queries/q79.sql. Data at https://github.com/ciioprof0/tackle/blob/main/data/processed/7_9.csv

10. Write a subquery using the NOT NULL operator in the inner query. Show the results and sort the results by key field(s). Interpret the output.

The results show that all reels (reel_ids 1, 2, 3, 4, and 5) in the dataset are paired with baits that have a specified bait_presentation, indicating that these reels are used in combinations where the bait presentation is clearly defined and not missing. This suggests that the dataset regarding bait presentation information for these reels is complete, and there are no instances where a reel is used without a defined bait presentation. This consistency helps ensure that all reels are used effectively with well-documented bait types.

Figure 28

Subquery Using NOT NULL



Note. Code at https://github.com/ciioprof0/tackle/blob/main/db/schema/queries/q710.sql. Data at https://github.com/ciioprof0/tackle/blob/main/data/processed/7_10.csv