Week 2A Assignment- Video Explanation Transcript

Kiera Griffiths

Part 1: SQL database set up (in DB Browser for SQLite)

1. Table Cleanup

```
DROP TABLE IF EXISTS ratings;
DROP TABLE IF EXISTS shows;
DROP TABLE IF EXISTS participants;
```

Deletes existing tables for three tables “ratings”, “shows”, and “participants”, if they already exist in the dataset and helps with reproducibility.

2. Create shows Table

```
CREATE TABLE shows (
  show_id INTEGER PRIMARY KEY AUTOINCREMENT,
  title TEXT NOT NULL
);
```

Creates a table called “shows” which uses item index show_id (integers) to track each show surveyed.

- show_id (item index)
    - Integer unique identifier
    - PRIMARY KEY enforces uniqueness
    - AUTOINCREMENT automatically assigns new IDs
- Title (metadata that is added later)
    - Name of the TV show
    - NOT NULL ensures every show has a valid label

3. Creating participants Table

```
CREATE TABLE participants (
  participant_id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
  name TEXT NOT NULL
);
```

Creates a table called “participants” which uses item index participant _id (integers) to track each show surveyed.

4. Creating ratings Table

```
CREATE TABLE ratings (
  participant_id INTEGER,
  show_id INTEGER,
  rating INTEGER CHECK (rating BETWEEN 1 AND 5),
  PRIMARY KEY (participant_id, show_id),
  FOREIGN KEY (participant_id) REFERENCES participants(participant_id),
  FOREIGN KEY (show_id) REFERENCES shows(show_id)
);
```

Creates a table called “ratings” which references the shows and participants tables and IDs (as foreign keys).

- **Stores the numeric rating**
- **CHECK enforces the 1–5 scale**
- **Prevents invalid data at the database level**

5. Naming Shows

```
INSERT INTO shows (title) VALUES
('Bridgerton'),
('Peaky Blinders'),
('Gilded Age'),
('Underground'),
('Shogun'),
('Warriors');
```

Lists actual six shows into shows database. Each show will be given an ID which it will use going forward in place of show title.

6. Inserting Ratings (Observed Preferences)

```
INSERT INTO ratings (participant_id, show_id, rating) VALUES
(1, 1, 2),
(1, 2, 3),
(1, 3, 5),
(1, 4, 4),

(2, 1, 4),
(2, 3, 5),
(2, 6, 5),

(3, 2, 5),
(3, 3, 1),
(3, 4, 5),
(3, 5, 4),

(4, 1, 3),
(4, 3, 5),
(4, 4, 5),
(4, 5, 4),

(5, 3, 4),
(5, 4, 5),
(5, 6, 5);
```

Participant ratings in (participant, show, rating) order. If a show is not rated by a participant, the row for that show is left off the list.

Part 2- Transfer SQL data to R

1. Load the DBI Package

```
library(DBI)
```

Loads the DBI package, connecting to SQL database from R.

2. Load the RSQLite Package

```
library(RSQLite)
```

Loads the SQLite driver, enabling R to connect to SQLite database.

3. Load the dplyr Package

```
library(dplyr)
```

Loads dplyr, a package for readable data manipulation that joins tables, summarizes ratings, and groups data by shows.

4. Connect to the Database

```
con <- dbConnect(SQLite(), "historical_tv_dramas_ratings.db")
```

Creates "con" a connection object that links R to the SQLite database file.

5. Load SQL Tables into R

```
shows <- dbReadTable(con, "shows")
participants <- dbReadTable(con, "participants")
ratings <- dbReadTable(con, "ratings")
```

Reads three SQL tables into R as data frames.

- **shows becomes a dataframe of items**
- **participants becomes a dataframe of users**
- **ratings becomes a dataframe of observed ratings**

6. Build Matrix

```
full_data <- participants %>%
 tidyr::crossing(shows) %>%
 left_join(ratings, by = c("participant_id", "show_id"))
```

Creates every possible combination of 5 participants and 6 shows, including missing values. The matrix either shows a rating or "NA".

7. Compute Average Ratings Per Show

```
average_ratings <- full_data %>%
 group_by(title) %>%
 summarise(
  average_rating = mean(rating, na.rm = TRUE),
  number_of_ratings = sum(!is.na(rating))
 )
```

Calculates average rating and counts number of ratings for each show.

- **Ignores missing values (na.rm = TRUE)**
- **!is.na(rating) excludes missing values**

	title	average_rating	number_of_ratings
1	Bridgerton	3.0	3
2	Gilded Age	4.0	5
3	Peaky Blinders	4.0	2
4	Shogun	4.0	2
5	Underground	4.75	4
6	Warriors	5.0	2

Conclusion: Missing ratings made shows' average rating highly sensitive to each rating they got. "Warriors" only had 2 ratings but a perfect 5.0 average rating. Meanwhile, "Gilded Age", the most popular show with 5 ratings, had a 4.0 average rating.