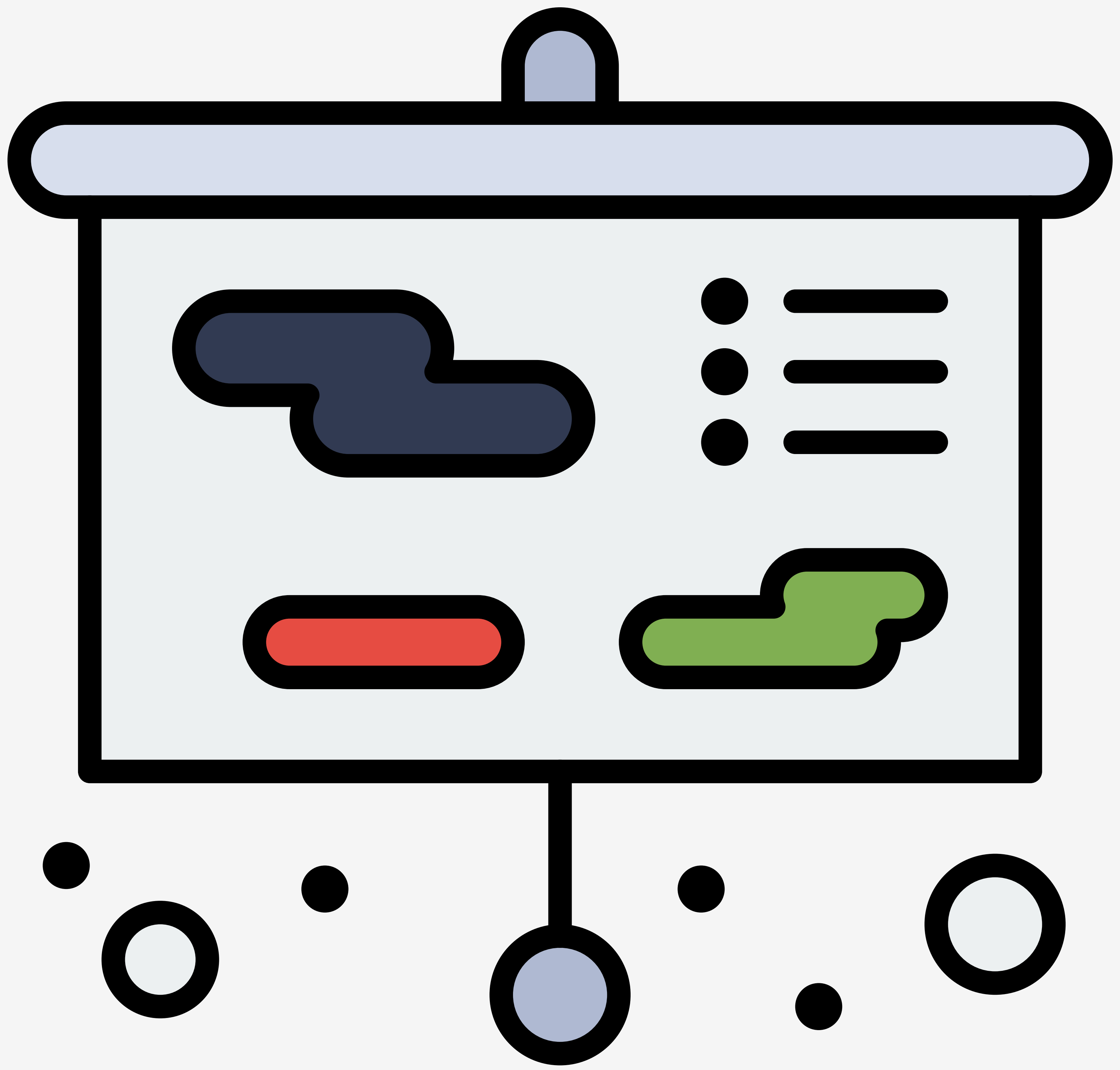
SERVER-SIDE WEB DEVELOPMENT

Lecture 4

TODAY'S TOPICS

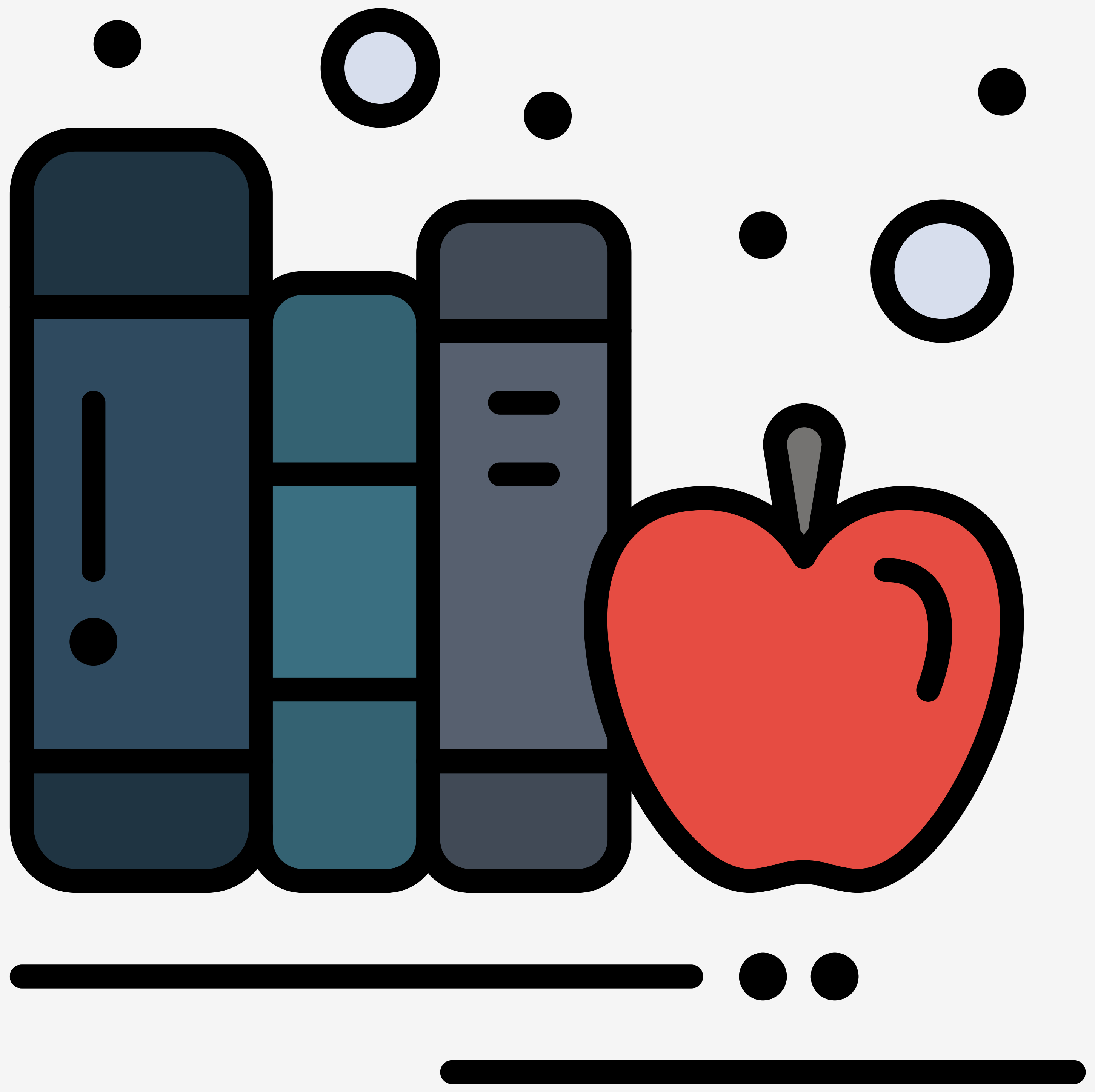


- Databases
- SQL Basics
- Participation: Movie Mayhem I
- Participation: Hybrid #2
- Exercise: Seussology DB I

QUESTIONS

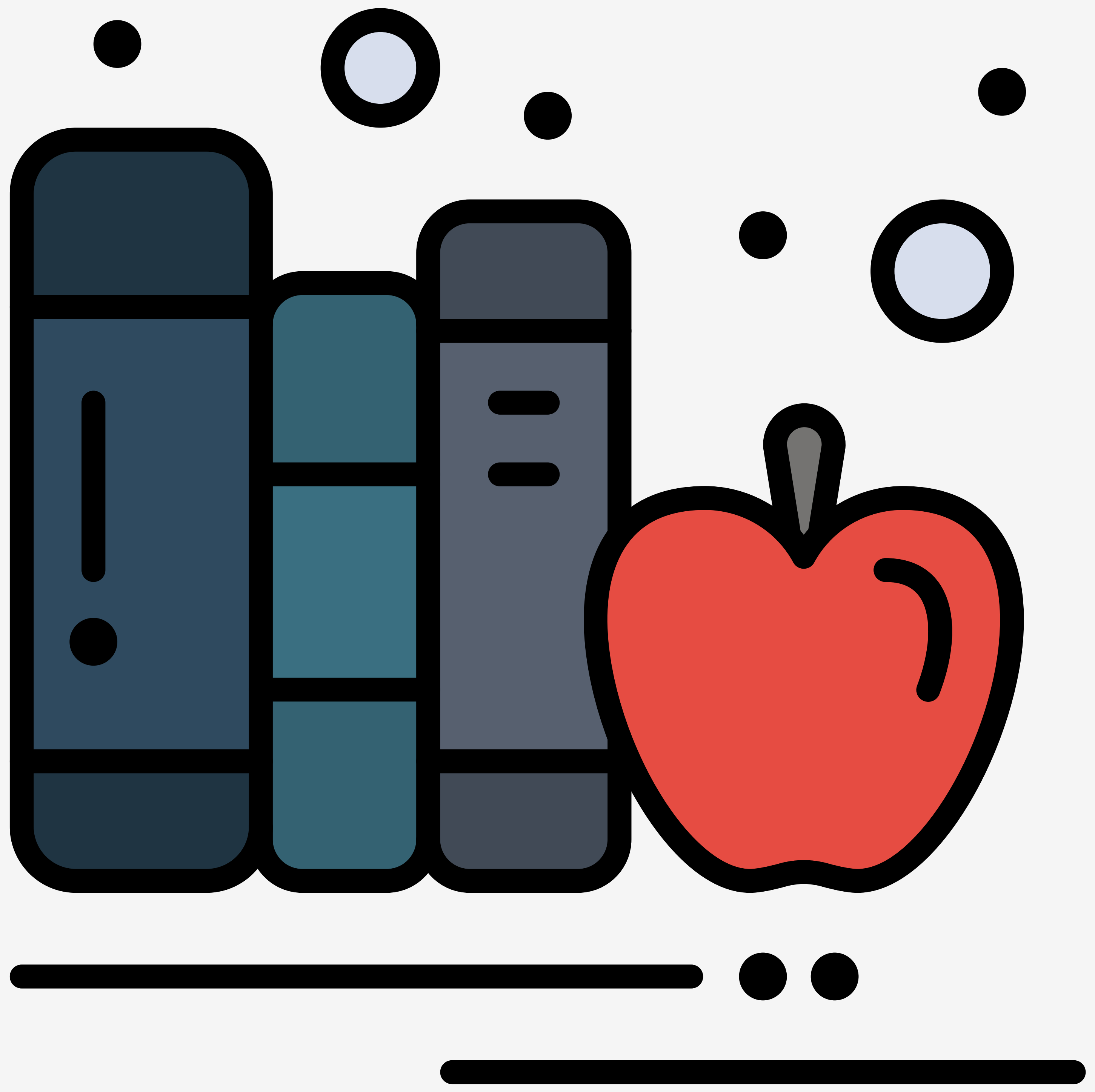
DATABASES

DATABASES



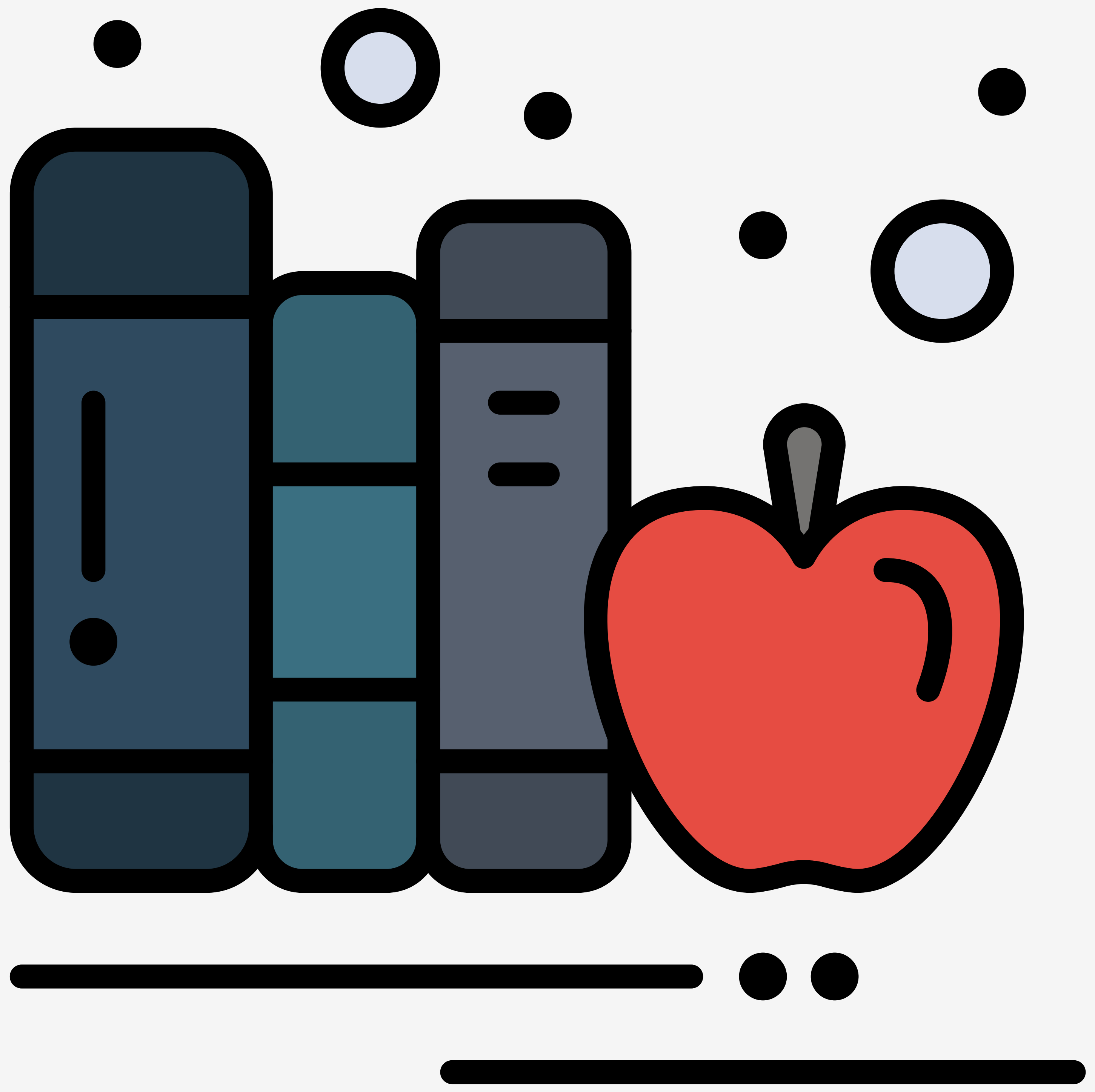
- A database is an organized collection of data managed by a database management system
- Advantages of using a database
  - Better scalability
  - Easier data management
  - Better accuracy
  - Better security
  - Better data integrity

DATABASE MANAGEMENT SYSTEMS



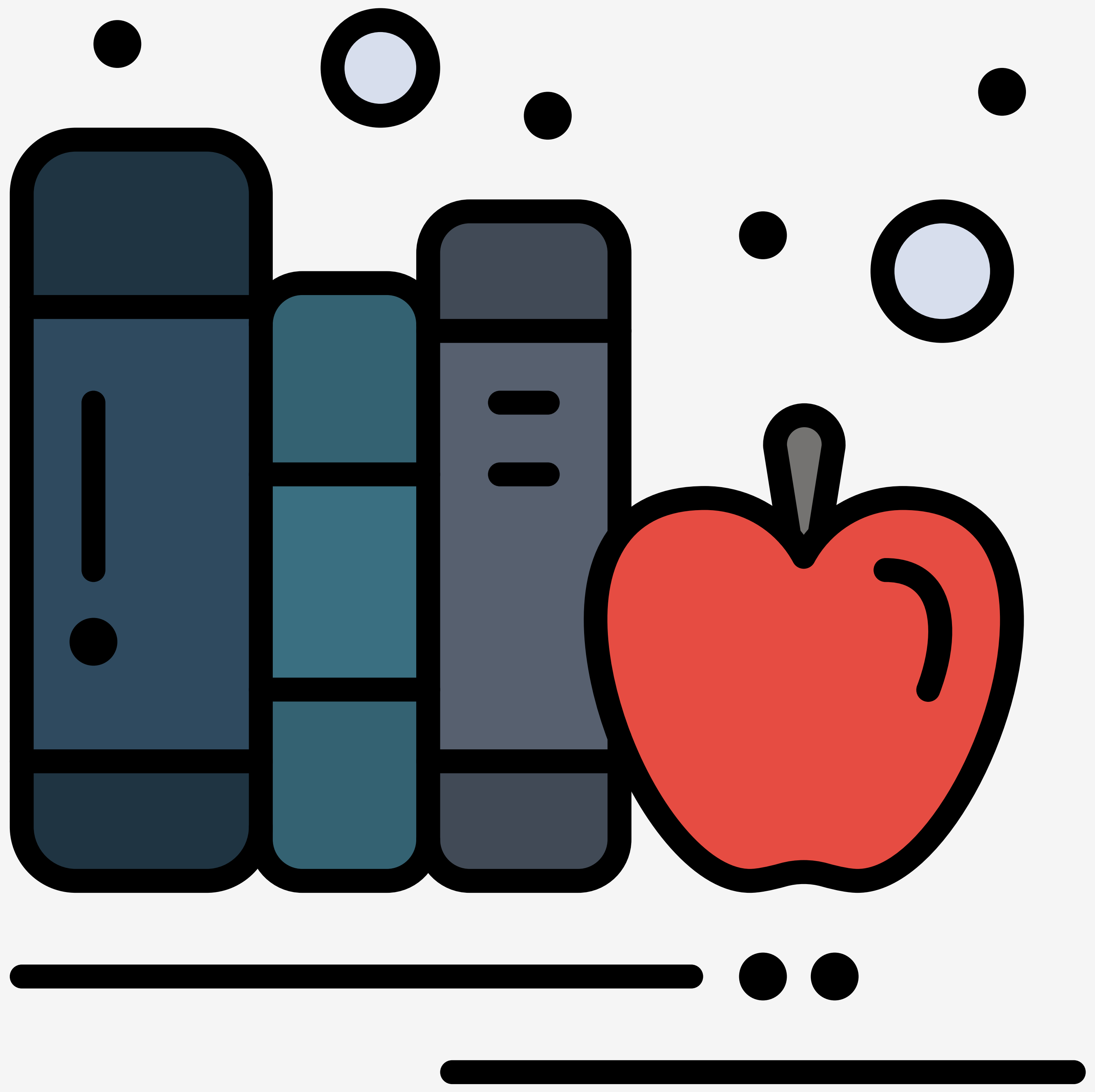
- A database management system (DBMS) is a program that opens database files and manages database tables
- A DBMS can manage multiple databases
- DBMS types
  - Relational
  - NoSQL

RELATIONAL DBMS



- All RDBMS have commons elements
  - tables, keys, relationships
  - Structure Query Language (SQL)
- Tables are like spreadsheets of data with columns and rows
- Keys are row identifiers, typically a number
- Relationships define how data is related
- These similarities makes moving from RDBMS easier for developers

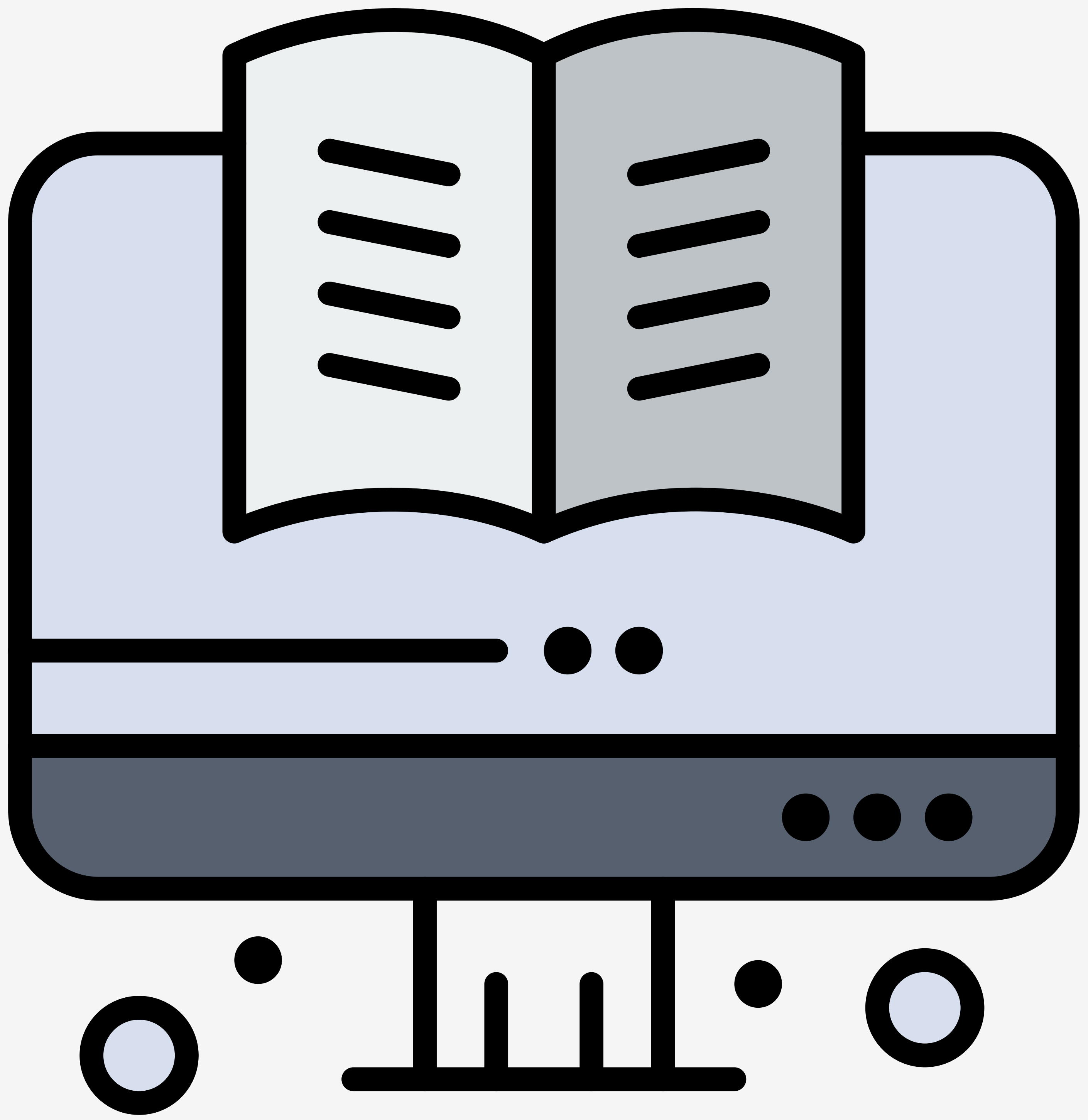
MYSQL



- MySQL is the most popular open-source RDBMS
- MySQL is a crucial part of the AMP stack.
- MySQL can be used by many languages
- MariaDB is a community driven, forked version of MySQL
- MariaDB remains almost fully compatible with MySQL

SQL BASICS

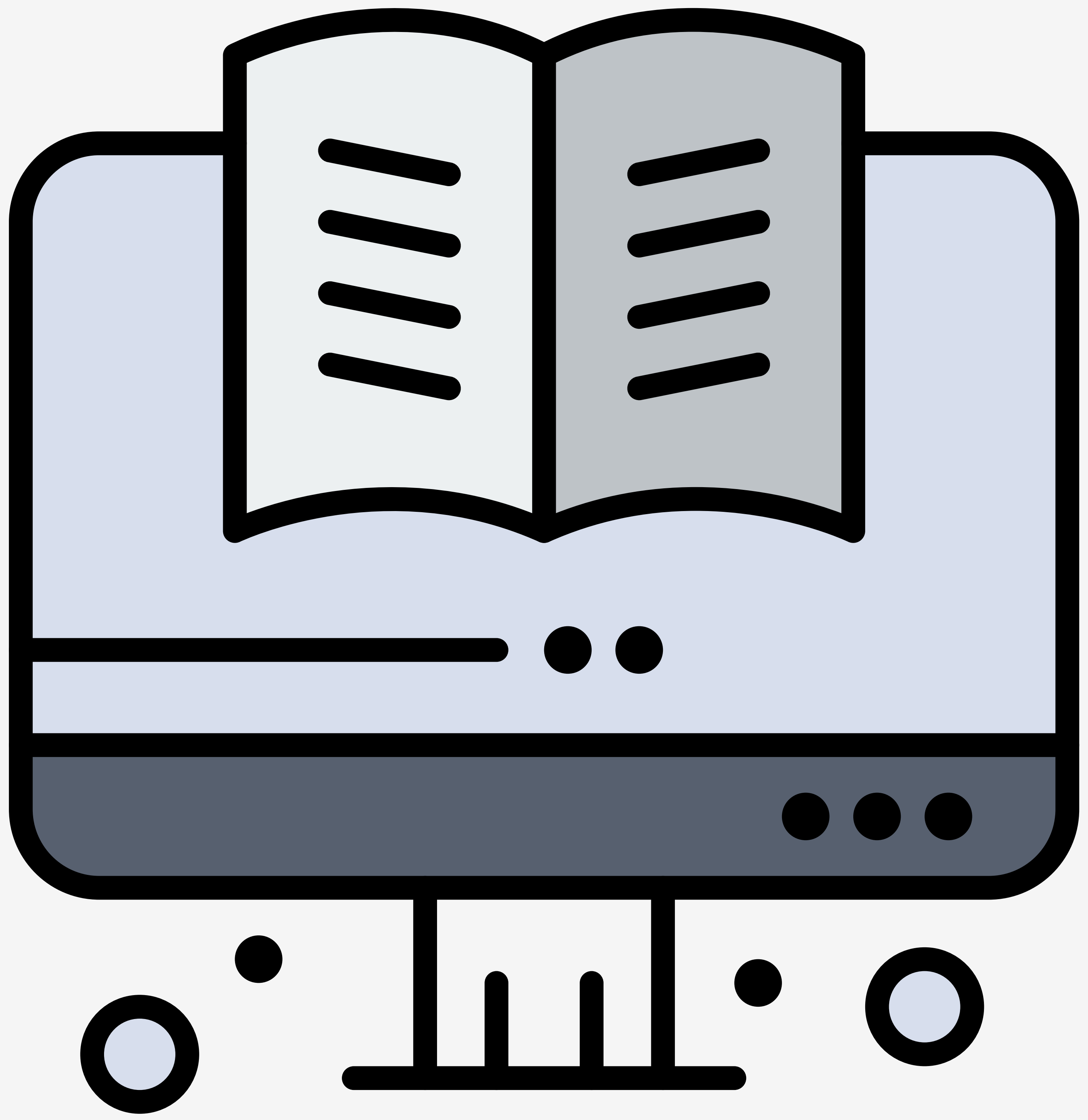
SQL



- Structured Query Language (SQL) is a scripting language for relational databases
- SQL is a declarative language with relatively few commands
- SQL complete four basic tasks
  - Create (`INSERT`)
  - Read (`SELECT`)
  - Update (`UPDATE`)
  - Delete (`DELETE`)

SELECT STATEMENTS

SELECT STATEMENTS



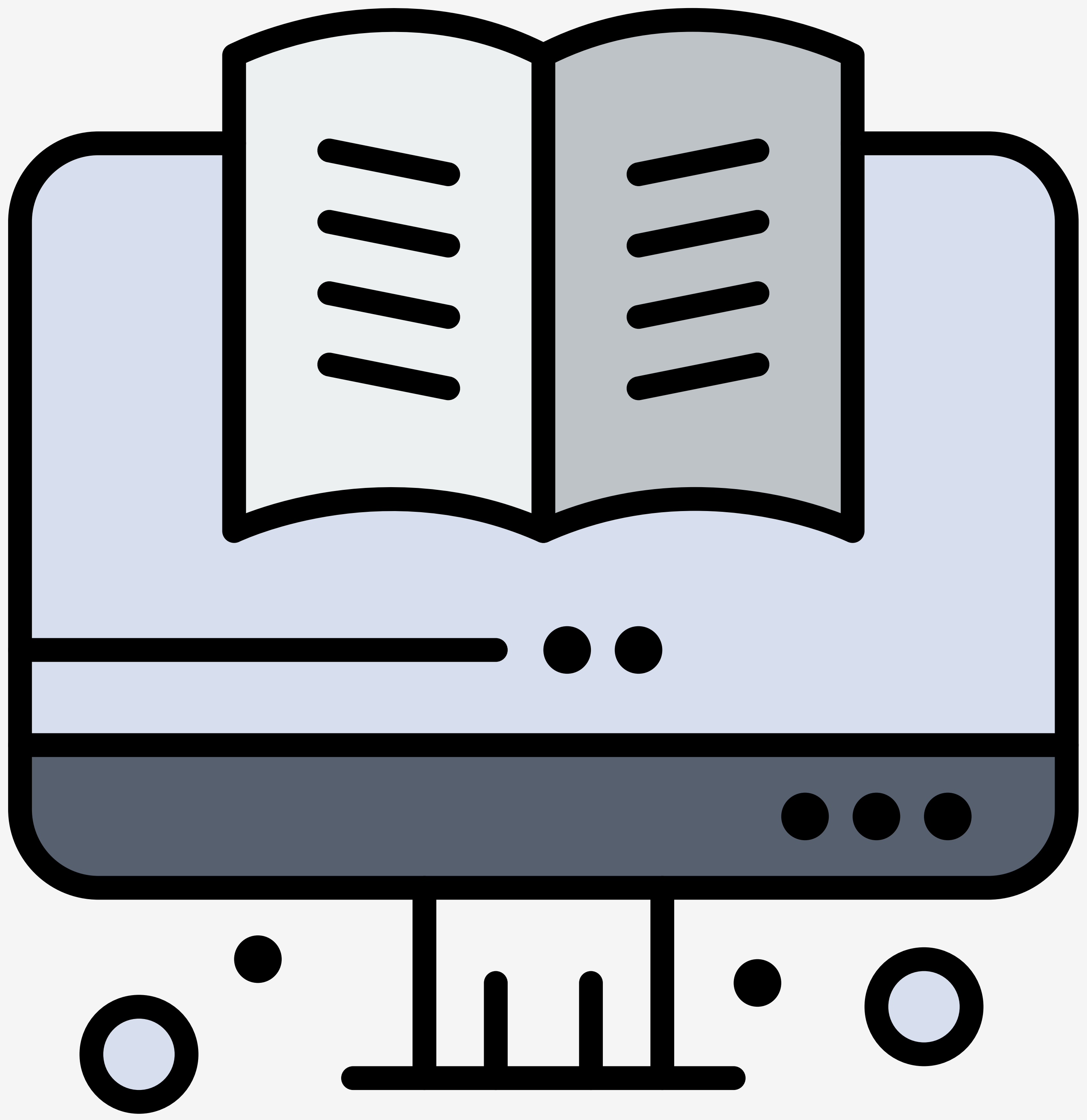
- The `SELECT` statement is ued to retrieve information from one or more tables
- `SELECT` *columns* `FROM` *table*

SELECT

```sql
-- Getting all movies with all columns
SELECT * FROM `movies`;

-- Getting all movies with `movie_title` and
`director` columns
SELECT `movie_title`, `director` FROM `movies`;
```

FILTERING SELECT STATEMENTS



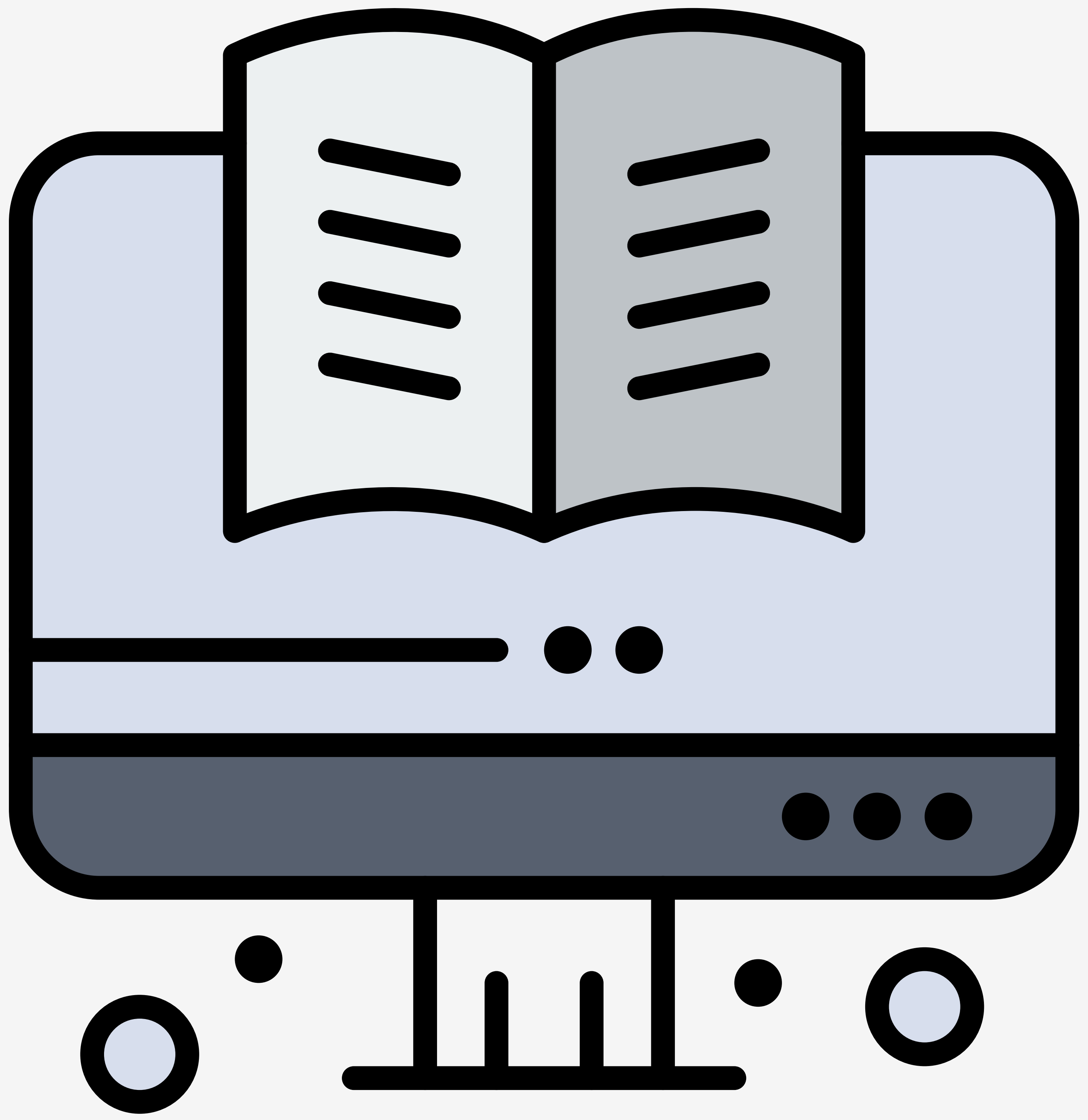
- The WHERE clause is used to filter rows
- The WHERE clause is followed by a condition that is tested against each row
- Only the rows that meet the condition are returned

WHERE

```sql
-- Getting all movies with an `movie_id` greater than 10
SELECT `movie_id`, `movie_title`, `director`
FROM `movies`
WHERE `movie_id` > 10;

-- Getting the movie with the `movie_title` of "Labyrinth"
SELECT `movie_id`, `movie_title`, `director`
FROM `movies`
WHERE `movie_title` = "Labyrinth";
```

USING WILDCARDS



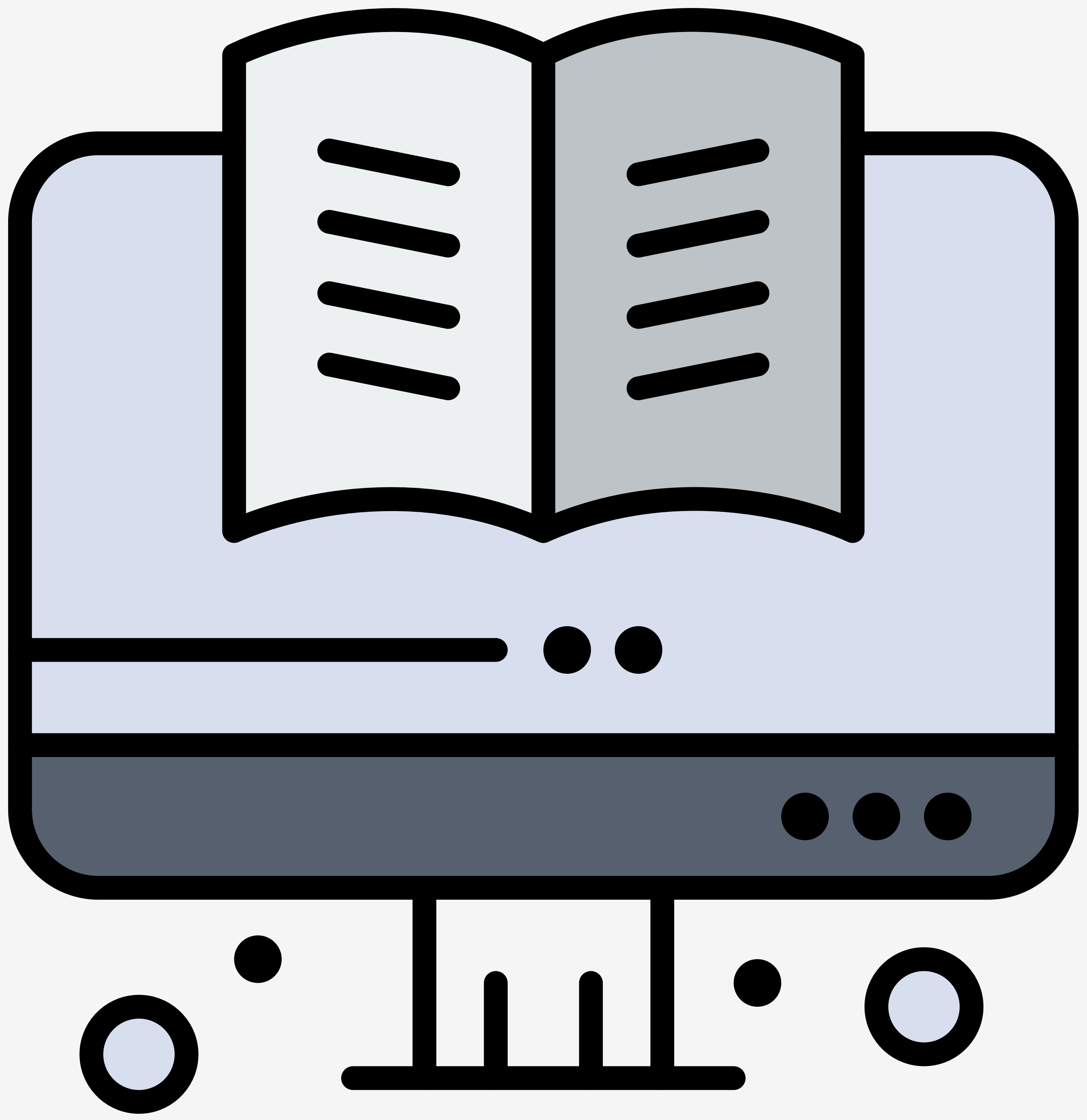
- The `LIKE` operator makes it possible to search for only part of a string
- A wildcard serves as a placeholder for one or more characters
    - `%` represents zero or more characters
    - `_` represents a single character

LIKE

```sql
-- Getting all movies whose `movie_title`
contains the letter 'a'
SELECT `movie_id`, `movie_title`, `director`
FROM `movies`
WHERE `movie_title` LIKE "%a%";

-- Getting all movies whose `movie_title`
starts with the letter 'G'
SELECT `movie_id`, `movie_title`, `director`
FROM `movies`
WHERE `movie_title` LIKE "G%";
```

COMPLEX FILTERS



- The AND and OR operators are used to filter records based on more than one condition
- The AND operator displays a record if all conditions are true
- The OR operator displays a record if any of the conditions are true

AND / OR

```sql
-- Getting all movies whose `movie_title` ends with the letter 't' AND  `director` starts with "Luc"
SELECT `movie_id`, `movie_title`, `director`
FROM `movies`
WHERE `movie_title` LIKE "%t"
AND `director` LIKE "Luc%";

-- Getting all movies whose `director` starts with "Luc" OR "Dean"
SELECT `movie_id`, `movie_title`, `director`
FROM `movies`
WHERE `director` LIKE "Luc%"
OR `director` LIKE "Dean%";
```

SORTING SELECT STATEMENTS

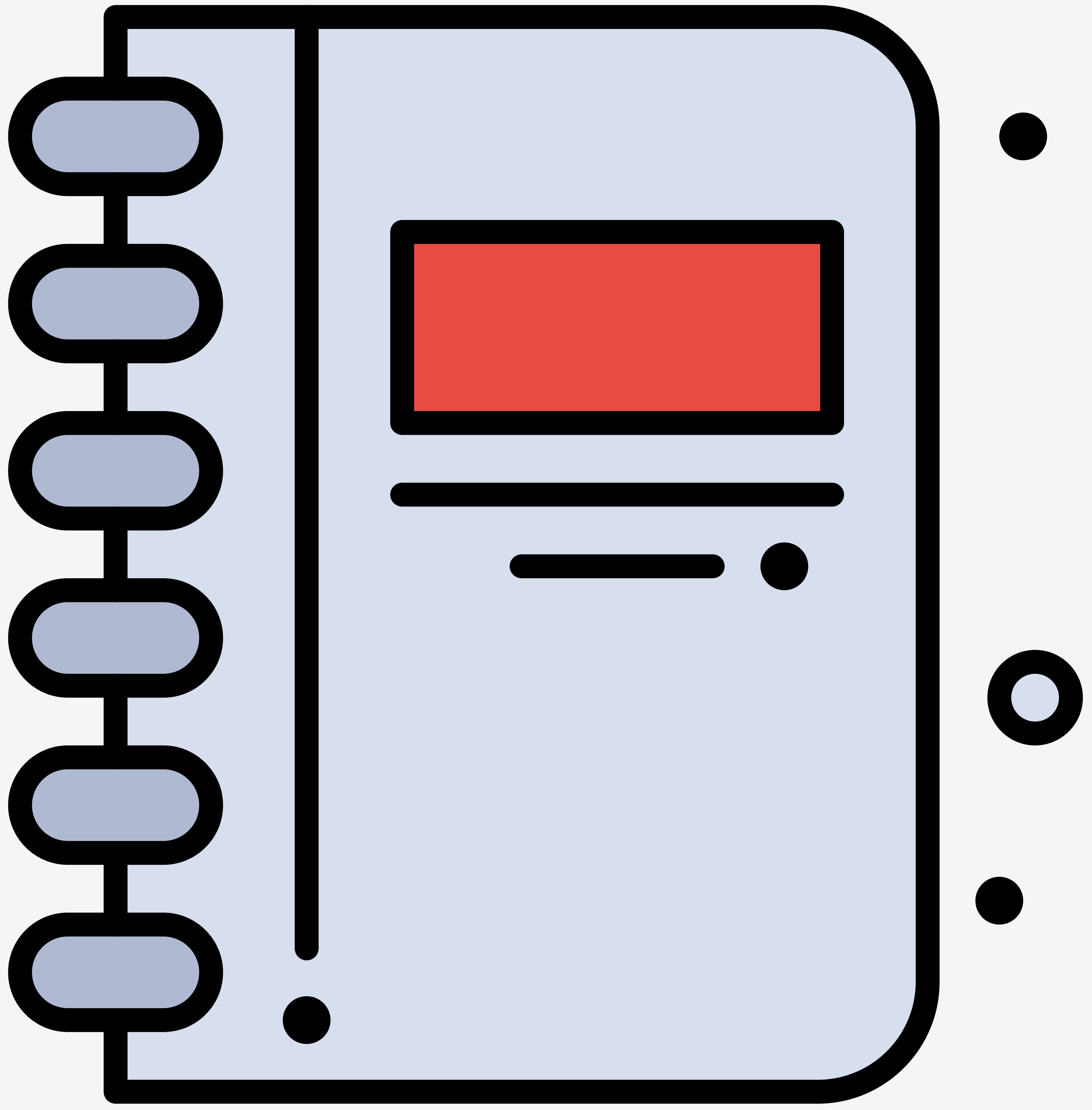
- The `ORDER BY` clause is used to sort the results
- The `ORDER BY` clause is followed by a list of columns to sort
- Columns can be sorted `ASC` or `DESC`

ORDER BY

```sql
-- Getting all movies sorted by the `year`
SELECT `movie_id`, `movie_title`, `director`,
`year`
FROM `movies`
ORDER BY `year`;

-- Getting all movies sorted by `movie_title`
in reverse alphabetical
SELECT `movie_id`, `movie_title`, `director`,
`year`
FROM `movies`
ORDER BY `movie_title` DESC;
```

LIMITING RESULTS



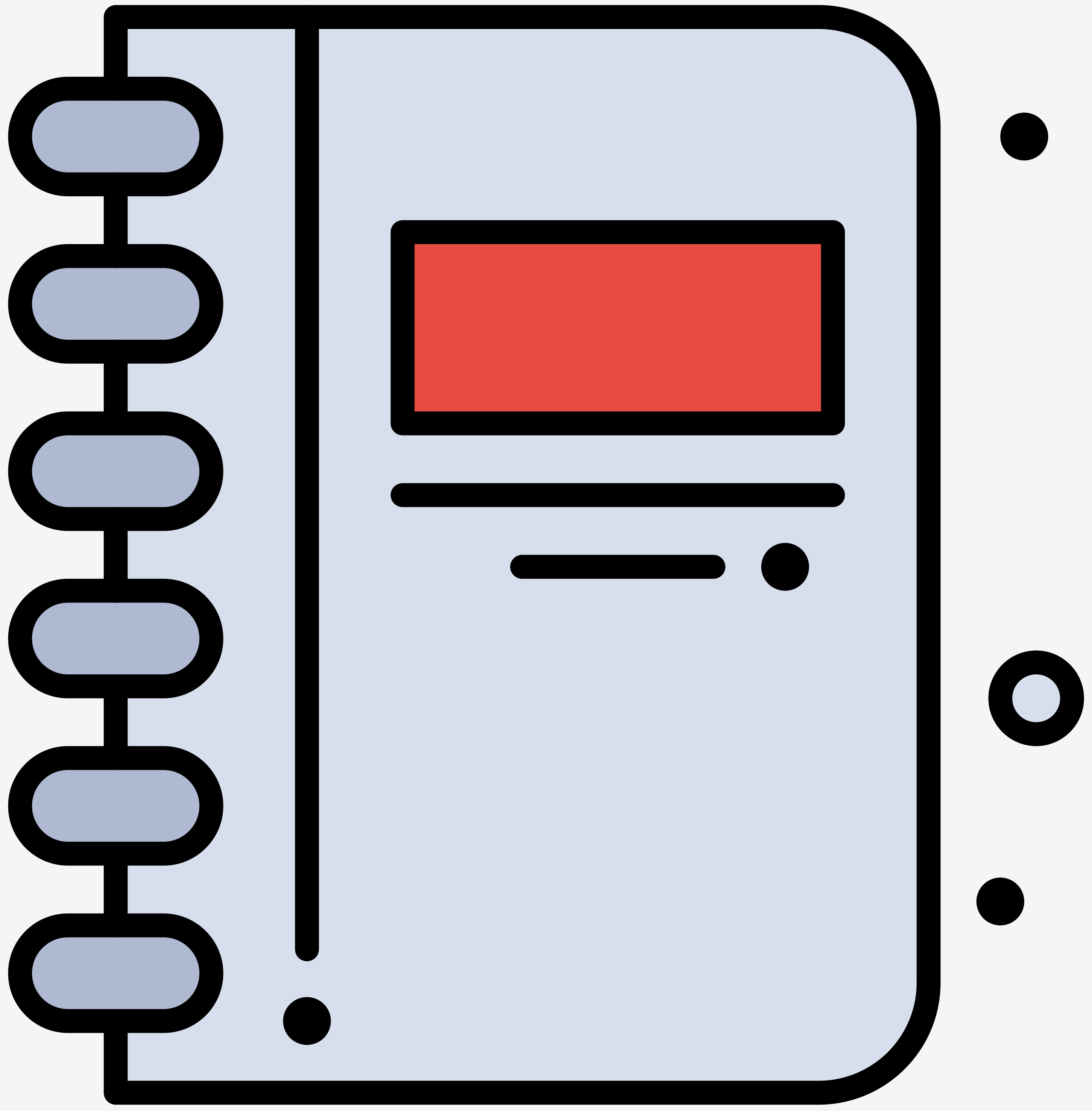
- The `LIMIT` clause can be used to set the number of rows to be returned
- The `LIMIT` clause can be used to paginate the results by add a second number
- The `LIMIT` clause can also be used with the `UDATE` and `DELETE` commands to limit the number of rows affected

LIMIT

```sql
-- Getting the 5 most recent movies
SELECT *
FROM `movies`
ORDER BY `year` DESC
LIMIT 5;

-- Getting the 5 most recent movies after the
10th movie
SELECT *
FROM `movies`
ORDER BY `year` DESC
LIMIT 10, 5;
```

FILTERING ON MULTIPLE VALUES



- The `IN` operator allows for multiple values to be specifies in a WHERE clause
- The `IN` operator can be used in place of multiple OR clauses on the same column

IN

```sql
-- Getting all movies from a list of years
SELECT *
FROM `movies`
WHERE `year` IN (1986, 1997, 1999, 2009);
```

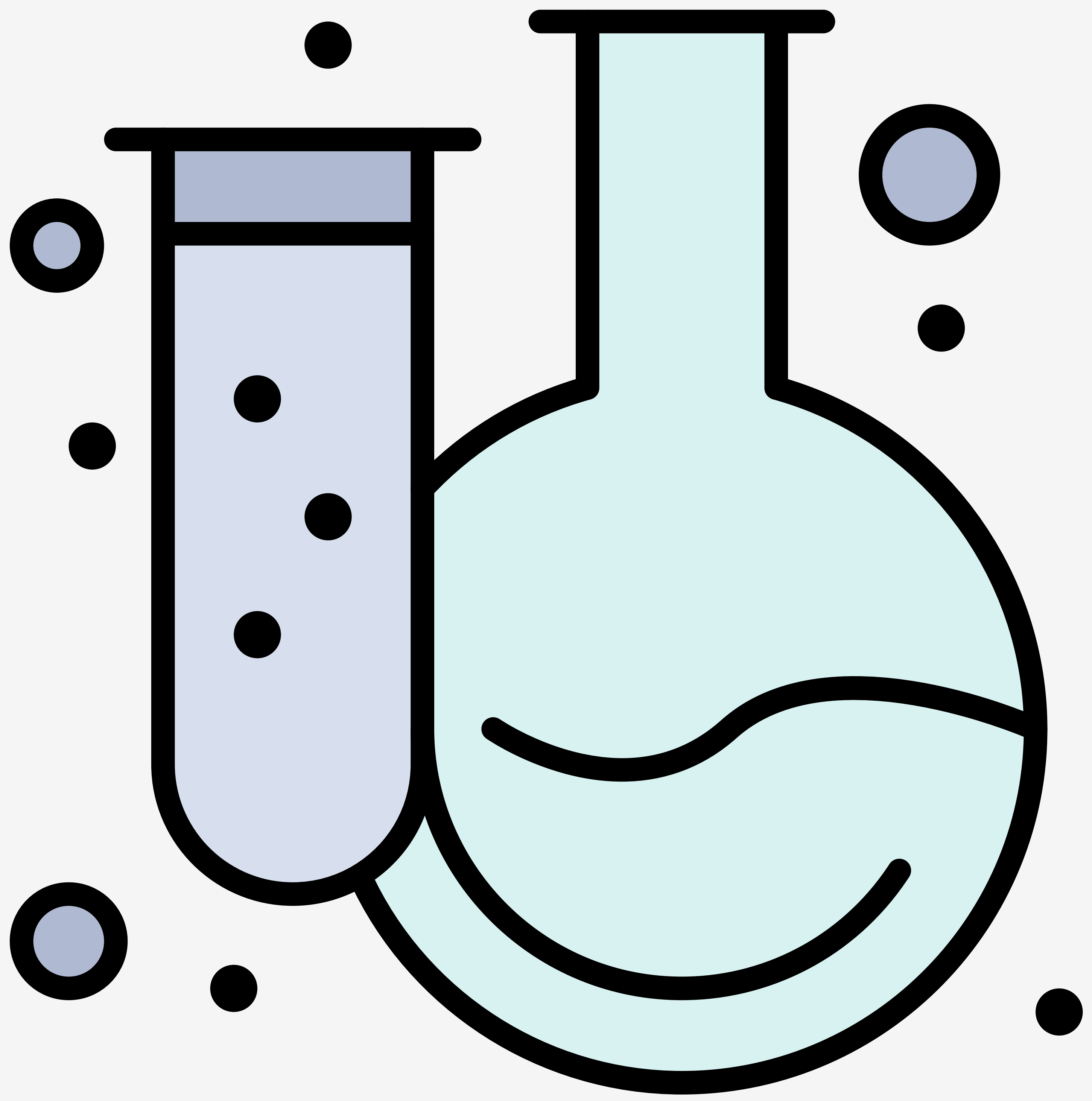
SUBQUERIES

- A subquery is a SELECT statement within another statement
- Subqueries can be use to get data from another table or from the same table

SUBQUERY

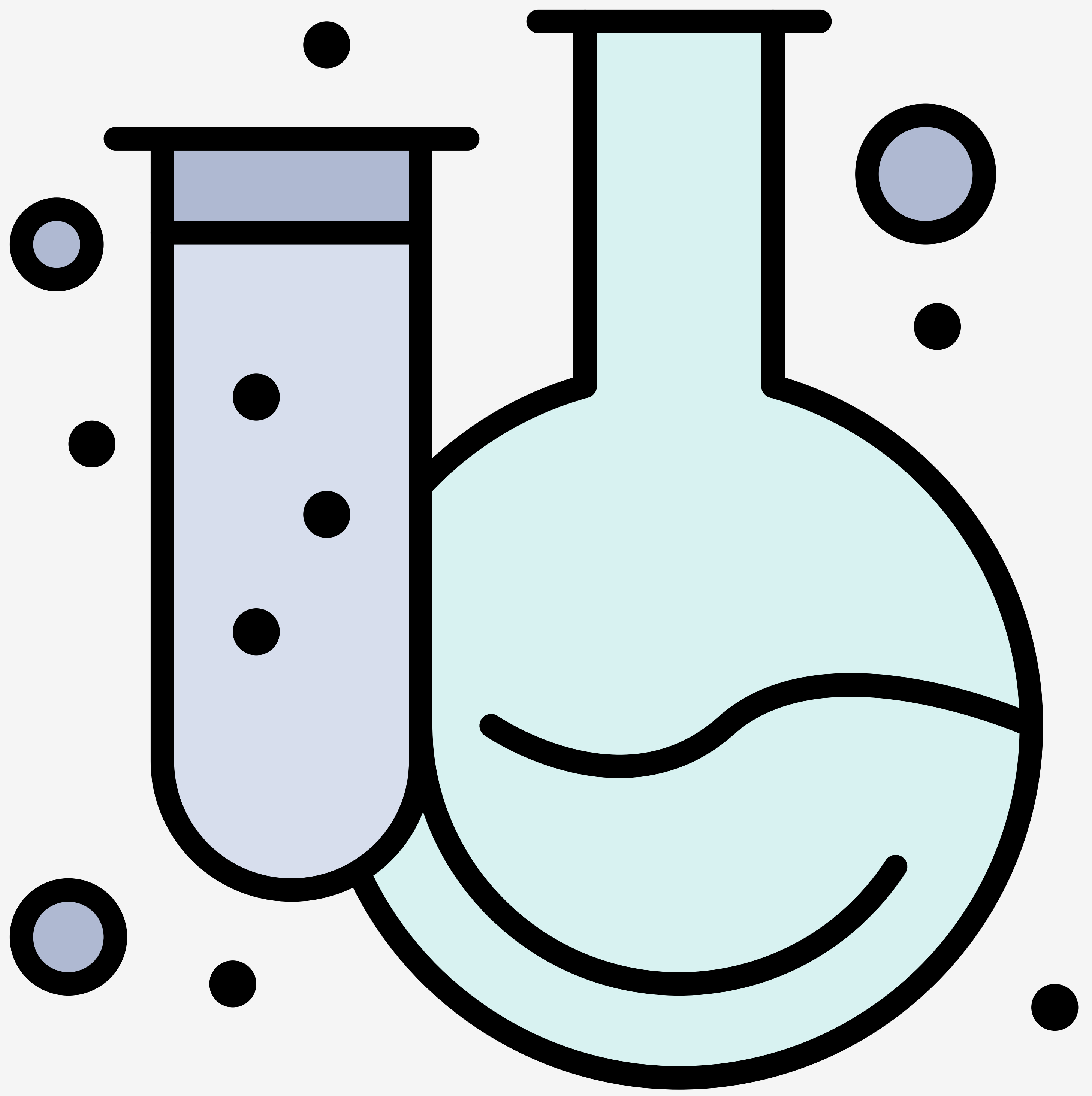
```
-- Getting all movies that have the "Fantasy"
genre
SELECT *
FROM `movies`
WHERE `genre_id` IN
  (SELECT `genre_id` FROM `genres` WHERE
    `genre_title` = 'Fantasy');
```

MOVIE MAYHEM I



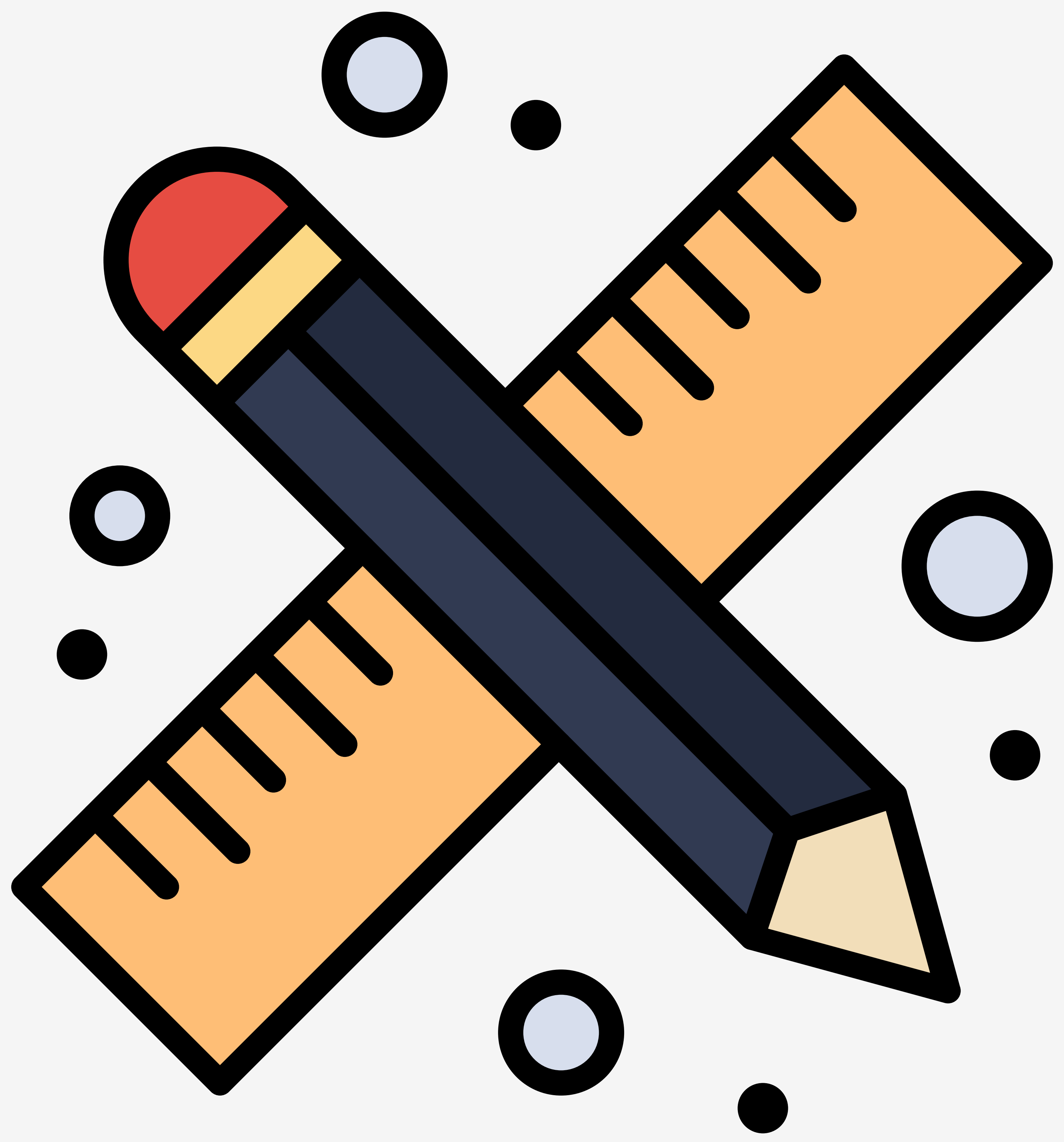
- *GITHUB CLASSROOM ASSIGNMENT*
- Import the Movie Mayhem DB
- Create *ONE* query for each task
- Use phpMyAdmin to run queries
- Save queries to `queries.sql`
- *DUE:* **Wed. Jun. 17 @ 11:59 PM**

HYBRID #2



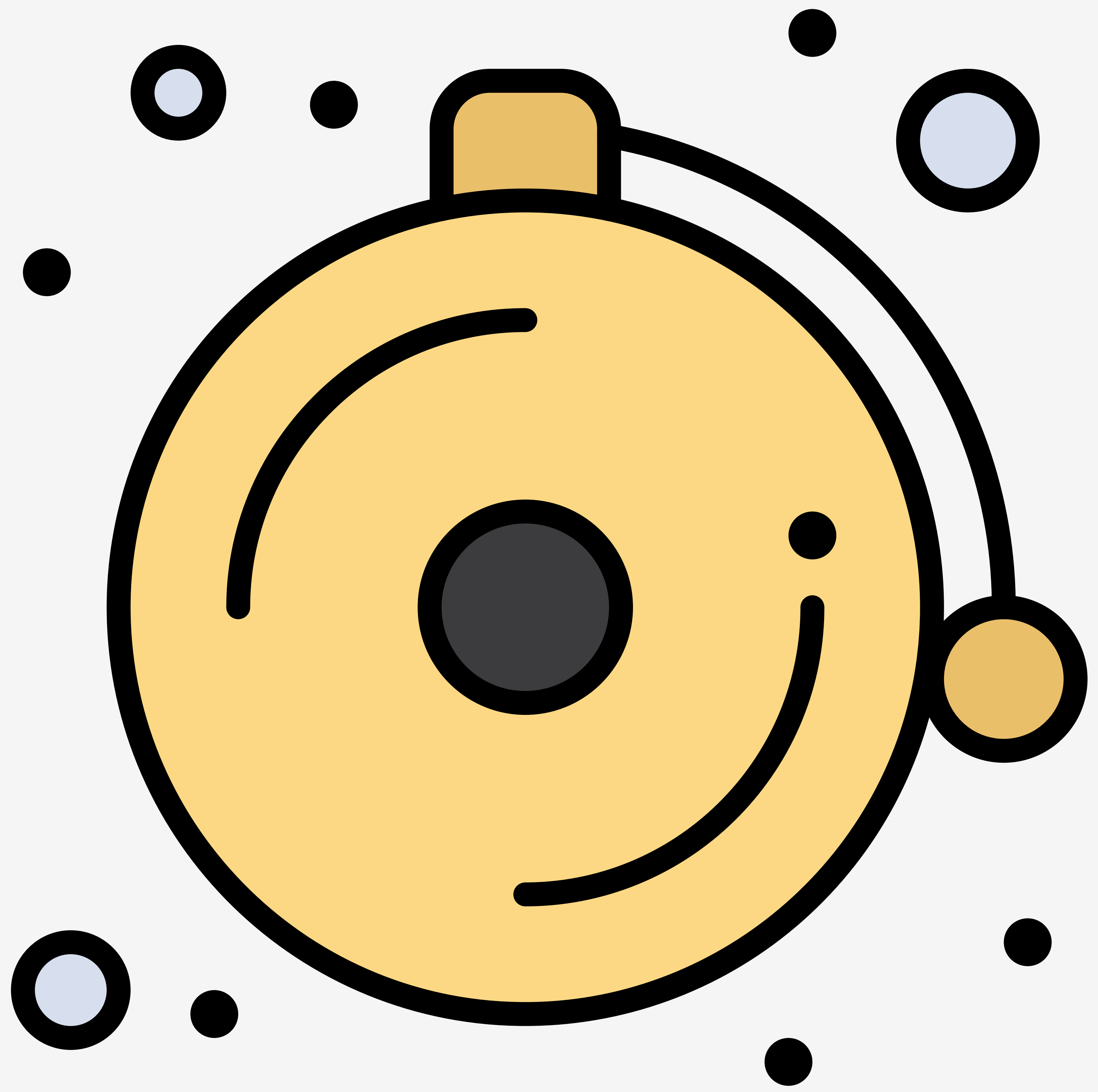
- Watch the entire course MySQL Essential Training (6 sections)
- Write 3 to 4 sentences for each section
- *DUE:* **Wed. Jun. 24 @ 11:59 PM**

SEUSSOLOGY DB I



- *GITHUB CLASSROOM ASSIGNMENT*
- Import the Seussology DB
- Create *ONE* query for each task
- Use phpMyAdmin to run queries
- Save queries to `queries.sql`
- *DUE:* **Wed. Jun. 24 @ 11:59 PM**

NEXT TIME...



- Non-SELECT Statements
- SQL Joins
- SQL Functions
- Participation: Movie Mayhem II
- Exercise: Seussology DB II