Odpowiedzi:

- DDL

1. CREATE DATABASE reading_room character set utf8mb4 collate utfmb4_polish_ci;

2. CREATE TABLE books (title VARCHAR(128), publish_date DATE, author VARCHAR(128), ISBN CHAR(16), category VARCHAR(256), pages SMALLINT, publisher VARCHAR(128), netto_price DECIMAL(5,2), vat_rate TINYINT, reserved TINYINT(1));

3. CREATE TABLE orders (order_number INT, first_name CARCHAR(32), last_name VARCHAR(32), order_date DATE, return_date DATE, book_title VARCHAR(128));

- DCL

1. CREATE USER 'reading_room_admin'@'localhost' IDENTIFIED BY 'reading_room_admin';

   GRANT ALL PRIVILEGES ON reading_room.* TO 'reading_room_admin'@'localhost';

2. \connect –mysqlx reading_room_admin@localhost;

   USE reading_room;

3. CREATE USER 'reading_room_user'@'localhost' IDENTIFIED BY 'reading_room_user';

   GRANT INSERT ON reading_room.* TO 'reading_room_user'@'localhost';

- Normalizacja

1. ALTER TABLE books drop column author;

   ALTER TABLE books DROP COLUMN category;
   ALTER TABLE books ADD COLUMN book_id INT PRIMARY KEY AUTO_INCREMENT;
   CREATE TABLE categories (category_id INT PRIMARY KEY AUTO_INCREMENT, name VARCHAR(32));
   CREATE TABLE authors (author_id INT PRIMARY KEY AUTO_INCREMENT, first_name VARCHAR(32), last_name VARCHAR(32));
   CREATE TABLE books_authors (book_id INT NOT NULL, author_id INT NOT NULL, PRIMARY KEY(book_id, author_id), FOREIGN KEY(book_id) REFERENCES books(book_id), FOREIGN KEY(author_id) REFERENCES authors(author_id));

   CREATE TABLE books_categories (book_id INT NOT NULL, category_id INT NOT NULL, PRIMARY KEY(book_id, categor_id), FOREIGN KEY(book_id) REFERENCES books(book_id), FOREIGN KEY(category_id) REFERENCES categories(category_id));

2. CREATE TABLE readers (reader_id INT PRIMARY KEY AUTO_INCREMENT, first_name VARCHAR(32), last_name VARCHAR(32));

```
ALTER TABLE orders DROP COLUMN first_name;
ALTER TABLE orders DROP COLUMN last_name;
ALTER TABLE orders ADD COLUMN order_id INT PRIMARY KEY
AUTO_INCREMENT;
ALTER TABLE orders DROP COLUMN book_title;
ALTER TABLE orders DROP COLUMN return_date;
ALTER TABLE orders DROP COLUMN order_date;
CREATE TABLE books_orders (book_id INT NOT NULL, order_id
INT NOT NULL, order_date DATE, return_date DATE, PRIMARY
KEY(book_id, order_id), FOREIGN KEY(book_id) REFERENCES
books(book_id), FOREIGN KEY(order_id) REFERENCES
orders(order_id));
CREATE TABLE readers_orders (reader_id INT NOT NULL,
order_id INT NOT NULL, PRIMARY KEY(reader_id, order_id),
FOREIGN KEY (reader_id) REFERENCES readers(reader_id),
FOREIGN KEY(order_id) REFERENCES orders(order_id));
```

- DML

1. Pomijam
2. \source C:\...[ścieżka do pliku]\book_source.sql
   ```
   INSERT INTO books SELECT * FROM new_books;
   ```
3. ```
   UPDATE books SET category='Bazy danych' WHERE
   category='SQL';
   ```
4. ```
   DELETE FROM books WHERE category IS NULL;
   ```
5. ```
   DROP TABLE new_books;
   ```

- PL/SQL

1. ```
   delimiter //
   CREATE FUNCTION overdue(reader INT) returns TINYINT READS
   SQL DATA
       BEGIN
           DECLARE temp INT;
           SELECT COUNT(*) INTO temp FROM readers r INNER JOIN
   readers_orders ro ON r.reader_id = ro.reader_id INNER JOIN
   orders o ON ro.order_id = o.order_id INNER JOIN books_orders
   bo ON o.order_id = bo.order_id INNER JOIN books bk on
   bo.book_id = bk.book_id WHERE r.reader_id=reader AND
   DATEDIFF(CURDATE(), bo.return_date) > 0;
           RETURN temp;
       END//
   delimiter ;
   ```
2. ```
   delimiter //
   CREATE PROCEDURE discount(IN percentage TINYINT,
   desired_book_isbn CHAR(17))
       BEGIN
            UPDATE books SET netto_price=(netto_price*(100-
   percentage))/100 WHERE ISBN=desired_book_isb
       END//
   delimiter ;
   ```
   ```

- DQL

1. Kopiujemy ścieżkę do katalogu test_db-master (C:/Users/…/test_db-master). Po otwarciu pliku employees.sql w linijkach od 112 pojawają się komendy source. Każdy plik z rozszerzeniem .dump należy rozbudować o skopiowaną ścieżkę. Uruchamiamy w MySQL Shell polecenie \source ze ścieżką do pliku employees.sql lub przez Wrokbench: Server -> Data Import -> Import From Self-Contained File i podajemy ścieżkę do employees.sql

2. `USE employees; SHOW TABLES;`

3. `SELECT * FROM employees ORDER BY hire_date ASC LIMIT 0,10;`

4. `SELECT SUM(salary) FROM salaries WHERE emp_no=111692;`

5. `SELECT CONCAT('Works for ', FLOOR(DATEDIFF(CURDATE(), e.hire_date)/365),' years') as 'Works for' FROM employees e WHERE e.emp_no = 38290;`

6. `SELECT e.first_name, e.last_name, s.salary from employees e inner join salaries s on e.emp_no = s.emp_no where s.salary > (select avg(si.salary) from salaries si inner join employees ei on si.emp_no = ei.emp_no inner join dept_emp de on ei.emp_no = de.emp_no where e.emp_no = de.emp_no);`

7. `select dm.emp_no, (select count(*) as subordinates from employees e left join dept_emp de on e.emp_no = de.emp_no group by (de.dept_no) having de.dept_no = dm.dept_no) as subordinates from dept_manager dm where dm.to_date > current_date();`

8. `select e.first_name, e.last_name from employees e join titles t on e.emp_no = t.emp_no group by e.emp_no having count(*) > 2;`

9. `select * from employees e natural join salaries s where (s.salary * 1.25 > 120000) and s.from_date between '1991-01-01' and '1991-12-31' and s.to_date between '1991-01-01';`

10. `select de.dept_no, count(*) as total_employees from dept_emp de group by de.dept_no having total_employees > 50000;`

11. `select d.dept_name from employees e natural join salaries s inner join dept_emp de on e.emp_no = de.emp_no inner join departments d on de.dept_no = d.dept_no group by de.dept_no order by avg(s.salary) desc limit 0,1;`

12. `select e.last_name, max(s.salary), de.dept_no from salaries s inner join dept_emp de on s.emp_no = de.emp_no inner join employees e on s.emp_no = e.emp_no where s.from_date between '1995-01-01' and '1995-12-31'and s.to_date between '1995-01-01' and '1995-12-31' group by de.dept_no;`

13. `select concat(e.last_name, ' pracuje, jako ', d.dept_name) from employees e inner join dept_emp de on e.emp_no = de.emp_no left join departments d on de.dept_no = d.dept_no where e.emp_no = 41295;`