Task 1: Database Schema Design and Optimization

1. SQL scripts to create the tables.

```
#CREATE DATABASE DBsurveys;

use DBsurveys;

CREATE TABLE surveys (
   survey_id int AUTO_INCREMENT,
   survey_name varchar(255),
   created_at date,
   PRIMARY KEY (survey_id)
);
CREATE TABLE questions (
   question_id int AUTO_INCREMENT,
   survey_id int,
   question_text varchar(255),
   PRIMARY KEY (question_id),
   FOREIGN KEY (survey_id) REFERENCES surveys(survey_id)
);
CREATE TABLE users (
   user_id int AUTO_INCREMENT,
   user_name varchar(255),
   user_email varchar(255),
   PRIMARY KEY (user_id),
         UNIQUE KEY (user_email)
);
CREATE TABLE responses (
   response_id int AUTO_INCREMENT,
   user_id int,
   question_id int,
   response_text varchar(255),
   PRIMARY KEY (response_id),
   FOREIGN KEY (user_id) REFERENCES users(user_id),
   FOREIGN KEY (question_id) REFERENCES questions(question_id)
);
```

2. SQL scripts to insert sample data into the tables (at least 3 surveys, 5 questions, 30 responses).

```
use DBsurveys;
INSERT INTO surveys (survey_name, created_at) VALUES
('Parents Satisfaction with the School', '2024-01-15'),
('Students’ Opinions on the Curriculum', '2024-02-10'),
('Evaluation of the School Environment', '2024-03-05');

INSERT INTO questions (survey_id, question_text) VALUES
(1, 'How satisfied are you with the quality of education your child receives?'),
(1, 'How would you rate the effectiveness of the school’s communication with parents?'),
(2, 'How engaging do you find the current curriculum?'),
(2, 'How well do you think the curriculum prepares you for future studies?'),
(3, 'How satisfied are you with the cleanliness of the school facilities?'),
(3, 'How safe do you feel in the school environment?');
INSERT INTO users (user_name, user_email) VALUES
('John Smith', 'john.smith@example.com'),
('Mary Johnson', 'mary.johnson@example.com'),
('Charles Williams', 'charles.williams@example.com'),
('Anna Brown', 'anna.brown@example.com'),
('Louis Davis', 'louis.davis@example.com');
INSERT INTO responses (user_id, question_id, response_text) VALUES
(1, 1, 'Neutral'),
(1, 2, 'Very satisfied'),
(2, 1, 'Satisfied'),
(2, 2, 'Neutral'),
(3, 1, 'Very dissatisfied'),
(3, 2, 'Satisfied'),
(4, 1, 'Dissatisfied'),
(4, 2, 'Dissatisfied'),
(5, 1, 'Very satisfied'),
(5, 2, 'Satisfied'),
(1, 3, 'Satisfied'),
(1, 4, 'Satisfied'),
(2, 3, 'Very satisfied'),
(2, 4, 'Very dissatisfied'),
(3, 3, 'Neutral'),
(3, 4, 'Very satisfied'),
(4, 3, 'Satisfied'),
(4, 4, 'Neutral'),
(5, 3, 'Dissatisfied'),
(5, 4, 'Neutral'),
(1, 5, 'Very dissatisfied'),
(1, 6, 'Satisfied'),
(2, 5, 'Neutral'),
(2, 6, 'Dissatisfied'),
(3, 5, 'Dissatisfied'),
(3, 6, 'Very satisfied'),
(4, 5, 'Very satisfied'),
(4, 6, 'Neutral'),
(5, 5, 'Very satisfied'),
(5, 6, 'Dissatisfied');
```

3. SQL script to create indexes for optimizing queries.

```
use DBsurveys;

CREATE INDEX idx_survey_id ON questions(survey_id);

CREATE INDEX idx_user_id ON responses(user_id);
CREATE INDEX idx_question_id ON responses(question_id);
CREATE INDEX idx_user_question ON responses(user_id, question_id);
```

Task 2: Query Writing, Optimization, and Advanced Analysis

SQL scripts for all queries.

1. **Basic Queries:**

#Write a query to retrieve all responses for a given survey, including survey name, question text, and response text.;

.

#First query without indexes, the logic is we select the data we want and make a join into the tables and add the where statement to sort by survey;

```
SELECT s.survey_name, q.question_text, r.response_text
FROM  responses r JOIN
          questions q ON r.question_id = q.question_id JOIN
    surveys s ON q.survey_id = s.survey_id
WHERE s.survey_id = 3;
```

#Query with indexes;

```
SELECT s.survey_name, q.question_text, r.response_text
FROM  surveys s JOIN
    questions q ON s.survey_id = q.survey_id JOIN
```

```
     responses r ON q.question_id = r.question_id
WHERE  s.survey_id = 3;
```

2. Advanced Queries:

#Write a query to calculate the average score for each survey, grouped by survey name.;

```
SELECT s.survey_name, AVG(
                                 CASE
                                         WHEN r.response_text = 'Very satisfied' THEN 5
                                         WHEN r.response_text = 'Satisfied' THEN 4
                                         WHEN r.response_text = 'Neutral' THEN 3
                                         WHEN r.response_text = 'Dissatisfied' THEN 2
                                         WHEN r.response_text = 'Very dissatisfied' THEN 1
                                         ELSE NULL
                                 END
                         ) AS average_score
FROM surveys s JOIN
    questions q ON s.survey_id = q.survey_id JOIN
    responses r ON q.question_id = r.question_id
GROUP BY s.survey_name;
```

#In order to optimize this query we add a column for adding the response_score and the indexes ;

```
ALTER TABLE responses
ADD COLUMN response_score INT;
```

# We add an update to add the new score to the existing responses;
# (we can make this into the table of make a view for example);

```
UPDATE responses
SET response_score = CASE response_text
           WHEN 'Very satisfied' THEN 5
           WHEN 'Satisfied' THEN 4
           WHEN 'Neutral' THEN 3
           WHEN 'Dissatisfied' THEN 2
           WHEN 'Very dissatisfied' THEN 1
        END;
```

# now we modify the query , we remove the case and just use the function AVG directly into the resoinse_score;

```
SELECT s.survey_name, AVG(r.response_score) AS average_score
FROM responses r JOIN
         questions q ON r.question_id = q.question_id JOIN
    surveys s ON q.survey_id = s.survey_id
```

```
GROUP BY s.survey_name
ORDER BY average_score DESC;
```

#Write a query to find the top 3 users with the highest average response score across all surveys.;

```
SELECT u.user_name, AVG(
                                        CASE r.response_text
                                        WHEN 'Very satisfied' THEN 5
                                        WHEN 'Satisfied' THEN 4
                                        WHEN 'Neutral' THEN 3
                                        WHEN 'Dissatisfied' THEN 2
                                        WHEN 'Very dissatisfied' THEN 1
                                        END
                              ) AS average_score
FROM responses r JOIN
          users u ON r.user_id = u.user_id
GROUP BY u.user_name
ORDER BY average_score DESC
LIMIT 3;

# Here we apply the same logic as before, removed the case.;

SELECT u.user_name, AVG(r.response_score) AS average_score
FROM responses r JOIN
           users u ON r.user_id = u.user_id
GROUP BY u.user_name
ORDER BY average_score DESC
LIMIT 3;
```

#Write a query to determine the distribution of responses for each question in a specific survey (e.g., count of each response).;

```
SELECT q.question_id, q.question_text, r.response_text, COUNT(*) AS response_count
FROM questions q JOIN
           responses r ON q.question_id = r.question_id
WHERE q.survey_id = 2
GROUP BY q.question_id, q.question_text, r.response_text
ORDER BY q.question_id, r.response_text;

/*



*/
```

Task 3: Stored Procedures and Views

1. Stored Procedure:

- Write a stored procedure that calculates the score for a survey based on responses.

```
use DBsurveys;

DELIMITER //
CREATE PROCEDURE CalculateSurveyScore(IN surveyId INT)
BEGIN
#create some variable that we need;
   DECLARE totalScore INT DEFAULT 0;
   DECLARE responseCount INT DEFAULT 0;
   DECLARE surveyScore DECIMAL(5,2) DEFAULT 0.00;

#we calculate the sum of the responses and the number of reponses;
   SELECT SUM(response_score) AS totalScore, COUNT(*) AS responseCount
   INTO totalScore, responseCount
   FROM responses r JOIN
                  questions q ON r.question_id = q.question_id
   WHERE q.survey_id = surveyId;
#now we calculate the percent with the sum divide by number of response;
   IF responseCount > 0 THEN
      SET surveyScore = totalScore / responseCount;
   ELSE
      SET surveyScore = 0.00;
   END IF;
#and we call the survey_id and the variable surveyScore NOTE: we can chage the ID for the
NAME;
   SELECT surveyId AS survey_id, surveyScore AS average_score;
END //
DELIMITER ;

#we try the SP CalculateSurveyScore;
CALL CalculateSurveyScore(1);
```

- Use a simple scoring algorithm where each answer has a weight, and the score is the sum of weights.

```
use DBsurveys;

DELIMITER //
CREATE PROCEDURE CalculateSurveyweights(IN surveyId INT)
BEGIN\
#create a variable that we need to insert the sum;
   DECLARE totalScore INT DEFAULT 0;

#we calculate the sum of the responses;
   SELECT SUM(response_score) AS totalScore
   INTO totalScore
   FROM responses r JOIN
            questions q ON r.question_id = q.question_id
   WHERE q.survey_id = surveyId;

#and we call the survey_id and the variable surveyScore NOTE: we can chage the ID for the NAME;
   SELECT surveyId AS survey_id, totalScore AS total_score;
END //
DELIMITER ;

#we try the SP CalculateSurveyweights;
CALL CalculateSurveyweights(1);
```

2. View:

- Create a view that displays the survey name, question text, response text, and calculated score for each response.

```
CREATE VIEW SurveyDetails AS
#We select the columns that we need to show and made a join;
SELECT s.survey_name, q.question_text, r.response_text, r.response_score
FROM  surveys s JOIN
          questions q ON s.survey_id = q.survey_id JOIN
     responses r ON q.question_id = r.question_id;

#we try the view;
 SELECT * FROM SurveyDetails;
```