



CS 353 Term Project

Design Report

Section 1

Group 35

Group Members:

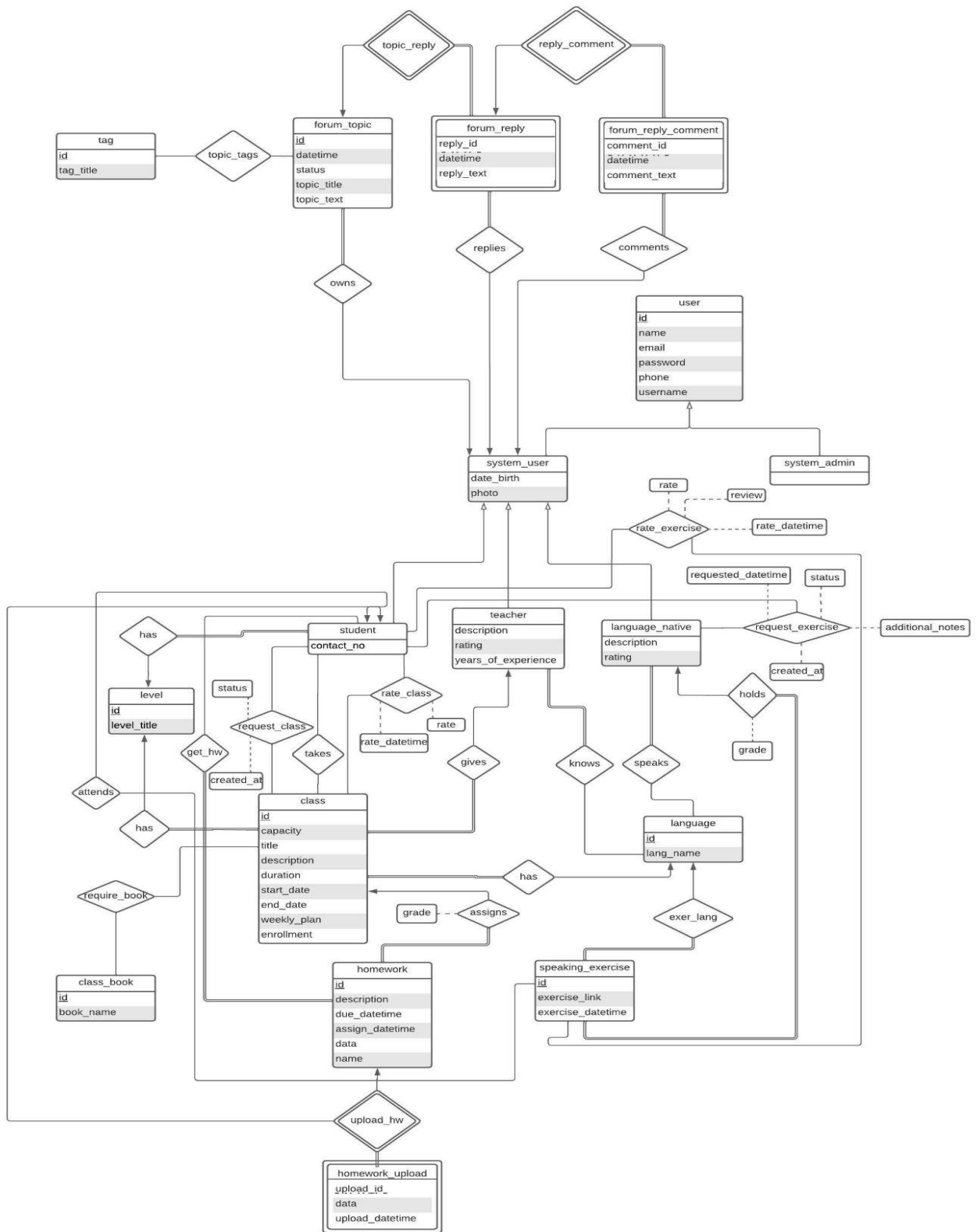
- | | |
|--------------------|----------|
| • Mert Şen | 21802602 |
| • Arda Atahan İbiş | 21901941 |
| • Şebnem Türkoğlu | 21901819 |

Topic: Online Language Learning Platform

Link: <https://online-language-learning-platform.vercel.app>

Link to ER Diagram: —————

Revised ER



Relational Models

1) User

Relational Schema:

user(id, name, email, password, phone, username, contact_no)

Candidate Keys:

Candidate Keys: id, email, username

Primary Key: id

Functional Dependencies:

id->{name, email, password, phone, username}

username->{id, name, email, password, phone}

email->{id, name, password, phone, username}

Normal Form:

BCNF since the left-hand sides of all of the functional dependencies are candidate keys.

SQL Creation:

```
DROP TABLE IF EXISTS user CASCADE;
CREATE TABLE user
(
    id SERIAL NOT NULL,
    name VARCHAR(50) NOT NULL,
    email VARCHAR(60) NOT NULL UNIQUE,
    password VARCHAR(50) NOT NULL,
    phone VARCHAR(20),
    username VARCHAR(50) NOT NULL UNIQUE,
    PRIMARY KEY (id)
);
```

2) System User

Relational Schema:

system_user(user_id, date_birth, photo)

Candidate Keys:

Candidate Key: user_id

Primary Key: user_id

Functional Dependencies:

user_id->{date_birth, photo}

Normal Form:

BCNF since the left-hand sides of all of the functional dependencies are candidate keys.

SQL Creation:

```
DROP TABLE IF EXISTS system_user CASCADE;
CREATE TABLE system_user
(
    user_id INT NOT NULL UNIQUE,
    date_birth DATE,
    photo TEXT,
    PRIMARY KEY (user_id),
    FOREIGN KEY (user_id) REFERENCES user(id) ON DELETE CASCADE
);
```

3) System Admin

Relational Schema:

system_admin(user_id)

Candidate Keys:

Candidate Key: user_id

Primary Key: user_id

Functional Dependencies:

There is no functional dependency.

Normal Form:

BCNF since there is no functional dependency.

SQL Creation:

```
DROP TABLE IF EXISTS system_admin CASCADE;
CREATE TABLE system_admin
(
    user_id INT NOT NULL UNIQUE,
    PRIMARY KEY (user_id),
    FOREIGN KEY (user_id) REFERENCES user(id) ON DELETE CASCADE
);
```

4) Student

Relational Schema:

student(user_id, contact_no, level_id)

FK: user_id references system_user(user_id)

FK: level_id references level(id)

Candidate Keys:

Candidate Key: user_id

Primary Key: user_id

Functional Dependencies:

user_id -> contact_no, level_id

Normal Form:

BCNF since user_id (the left-hand side of the functional dependency) is a candidate key.

SQL Creation:

```
DROP TABLE IF EXISTS student CASCADE;
CREATE TABLE student
(
    user_id INT NOT NULL UNIQUE,
    contact_no VARCHAR(12),
    level_id INT NOT NULL,
    PRIMARY KEY (user_id),
    FOREIGN KEY (user_id) REFERENCES system_user(user_id) ON DELETE
    CASCADE,
    FOREIGN KEY (level_id) REFERENCES level(id)
);
```

5) Teacher

Relational Schema:

teacher(user_id, description, rating, years_of_experience)

FK: user_id references system_user(user_id)

Candidate Keys:

Candidate Key: user_id

Primary Key: user_id

Functional Dependencies:

user_id -> {description, rating, years_of_experience}

Normal Form:

BCNF since user_id (the left-hand side of the functional dependency) is a candidate key.

SQL Creation:

```
DROP TABLE IF EXISTS teacher CASCADE;
CREATE TABLE teacher
(
    user_id INT NOT NULL UNIQUE,
    description TEXT,
    rating FLOAT,
    years_of_experience INT,
    PRIMARY KEY (user_id),
    FOREIGN KEY(user_id) REFERENCES system_user(user_id) ON DELETE
    CASCADE,
);
```

6) Language Native

Relational Schema:

language_native(user_id, description, rating)

FK: user_id references system_user(user_id)

Candidate Keys:

Candidate Key: user_id

Primary Key: user_id

Functional Dependencies:

user_id -> {description, rating}

Normal Form:

BCNF since user_id (the left-hand side of the functional dependency) is a candidate key.

SQL Creation:

```

DROP TABLE IF EXISTS language_native CASCADE;
CREATE TABLE language_native
(
    user_id INT NOT NULL UNIQUE,
    description TEXT,
    rating FLOAT,
    PRIMARY KEY (user_id),
    FOREIGN KEY (user_id) REFERENCES system_user (user_id) ON DELETE
CASCADE,
);

```

7) Class

Relational Schema:

class(id, capacity, title, description, duration, start_date, end_date, weekly_plan, enrollment, lang_id, teacher_id, level_id)

FK: lang_id references language(id)

FK: teacher_id references teacher(user_id)

FK: level_id references level(id)

Candidate Keys:

Candidate Key: id

Primary Key: id

Functional Dependencies:

id -> {capacity, title, description, duration, start_date, end_date, weekly_plan, enrollment, lang_id, teacher_id, level_id}

Normal Form:

BCNF since id (the left-hand side of the functional dependency) is a candidate key.

SQL Creation:

```

DROP TABLE IF EXISTS class CASCADE;
CREATE TABLE class
(
    id SERIAL NOT NULL,
    capacity INT,
    title VARCHAR(100),
    description TEXT,
    duration INT,

```

```

start_date DATE,
end_date DATE,
weekly_plan TEXT,
enrollment INT,
lang_id INT NOT NULL,
teacher_id INT NOT NULL,
level_id INT NOT NULL,
image BYTEA,
PRIMARY KEY (id),
FOREIGN KEY(lang_id) REFERENCES language(id),
FOREIGN KEY(teacher_id) REFERENCES teacher(user_id),
FOREIGN KEY(level_id) REFERENCES level(id)
);

```

8) Speaking Exercise

Relational Schema:

speaking_exercise(id, exercise_link, exercise_datetime, grade, student_id, lang_id, language_native_id)

FK: student_id references student(user_id)

FK: lang_id references language(id)

FK: language_native_id references language_native(user_id)

Candidate Keys:

Candidate Key: id

Primary Key: id

Functional Dependencies:

id -> {exercise_link, exercise_datetime, grade, student_id, lang_id, language_native_id}

Normal Form:

BCNF since id (the left-hand side of the functional dependency) is a candidate key.

SQL Creation:

```

DROP TABLE IF EXISTS speaking_exercise CASCADE;
CREATE TABLE speaking_exercise
(
    id SERIAL NOT NULL,
    exercise_link TEXT,

```



```

exercise_datetime TIMESTAMP,
grade FLOAT,
student_id INT NOT NULL,
lang_id INT NOT NULL,
language_native_id INT NOT NULL,
PRIMARY KEY (id),
FOREIGN KEY(student_id) REFERENCES student(user_id),
FOREIGN KEY(lang_id) REFERENCES language(id),
FOREIGN KEY(language_native_id) REFERENCES language_native(user_id)
);

```

9) Class Book

Relational Schema:

class_book(id, book_name)

Candidate Keys:

Candidate Key: id

Primary Key: id

Functional Dependencies:

id -> {book_name}

Normal Form:

BCNF since id (the left-hand side of the functional dependency) is a candidate key.

SQL Creation:

```

DROP TABLE IF EXISTS class_book CASCADE;
CREATE TABLE class_book
(
    id SERIAL NOT NULL,
    book_name TEXT,
    PRIMARY KEY (id),
);

```

10) Homework

Relational Schema:

homework(id, name, description, due_datetime, assign_datetime, grade, class_id)

FK: class_id references class(id)

Candidate Keys:

Candidate Key: id

Primary Key: id

Functional Dependencies:

id -> {name, description, due_datetime, assign_datetime, grade, class_id}

Normal Form:

BCNF since id (the left-hand side of the functional dependency) is a candidate key.

SQL Creation:

```
DROP TABLE IF EXISTS homework CASCADE;
CREATE TABLE homework
(
    id SERIAL NOT NULL,
    name VARCHAR(100),
    description TEXT,
    due_datetime TIMESTAMP,
    assign_datetime TIMESTAMP,
    grade FLOAT,
    class_id INT NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY(class_id) REFERENCES class(id)
);
```

11) Homework Upload

Relational Schema:

homework_upload(homework_id, upload_id, student_id, upload_datetime, data)

FK: homework_id references homework(id)

FK: student_id references student(user_id)

Candidate Keys:

Candidate Key: {homework_id, upload_id}

Primary Key: {homework_id, upload_id}

Functional Dependencies:

{homework_id, upload_id} -> {student_id, upload_datetime, data}

Normal Form:

BCNF since the left-hand side of the functional dependency is a candidate key.

SQL Creation:

```
DROP TABLE IF EXISTS homework_upload CASCADE;
CREATE TABLE homework_upload
(
    homework_id INT NOT NULL,
    upload_id INT NOT NULL,
    student_id INT NOT NULL,
    upload_datetime TIMESTAMP,
    data bytea,
    UNIQUE (homework_id, upload_id),
    PRIMARY KEY (homework_id, upload_id),
    FOREIGN KEY(homework_id) REFERENCES homework(id) ON DELETE CASCADE,
    FOREIGN KEY(student_id) REFERENCES student(user_id)
);
```

12) Language

Relational Schema:

language(id, lang_name)

Candidate Keys:

Candidate Keys: id, lang_name

Primary Key: id

Functional Dependencies:

id -> {lang_name}

lang_name -> {id}

Normal Form:

BCNF since the left-hand sides of all of the functional dependencies are candidate keys.

SQL Creation:

```
DROP TABLE IF EXISTS language CASCADE;
```

```
CREATE TABLE language
(
    id SERIAL NOT NULL,
    lang_name VARCHAR(80) NOT NULL UNIQUE,
    PRIMARY KEY (id)
);
```

13) Level

Relational Schema:

level(id, level_title)

Candidate Keys:

Candidate Keys: id, level_title

Primary Key: id

Functional Dependencies:

id -> {level_title}

level_title -> {id}

Normal Form:

BCNF since the left-hand sides of all of the functional dependencies are candidate keys.

SQL Creation:

```
DROP TABLE IF EXISTS level CASCADE;
CREATE TABLE level
(
    id SERIAL NOT NULL,
    level_title VARCHAR(80) NOT NULL UNIQUE,
    PRIMARY KEY (id)
);
```

14) Forum Topic

Relational Schema:

forum_topic(id, datetime, status, topic_title, topic_text, user_id)

FK: user_id references system_user(user_id)

Candidate Keys:

Candidate Key: id

Primary Key: id

Functional Dependencies:

id -> {datetime, status, topic_title, topic_text, user_id}

Normal Form:

BCNF since id (the left-hand side of the functional dependency) is a candidate key.

SQL Creation:

```
DROP TABLE IF EXISTS forum_topic CASCADE;
CREATE TABLE forum_topic
(
    id SERIAL NOT NULL,
    datetime TIMESTAMP,
    status VARCHAR(20),
    topic_title TEXT,
    topic_text TEXT,
    user_id INT NOT NULL,
    PRIMARY KEY (id),
    FOREIGN KEY (user_id) REFERENCES system_user (user_id)
);
```

15) Forum Reply

Relational Schema:

forum_reply(reply_id, topic_id, datetime, reply_text, user_id)

FK: user_id references system_user(user_id)

FK: topic_id references forum_topic(id)

Candidate Keys:

Candidate Key: {reply_id, topic_id}

Primary Key: {reply_id, topic_id}

Functional Dependencies:

{reply_id, topic_id} -> {datetime, reply_text, user_id}

Normal Form:

BCNF since the left-hand side of the functional dependency is a candidate key.

SQL Creation:

```
DROP TABLE IF EXISTS forum_reply CASCADE;
CREATE TABLE forum_reply
(
    reply_id INT NOT NULL,
    datetime TIMESTAMP,
    reply_text TEXT,
    user_id INT NOT NULL,
    topic_id INT NOT NULL,
    UNIQUE (reply_id, topic_id),
    PRIMARY KEY (reply_id, topic_id),
    FOREIGN KEY (user_id) REFERENCES system_user(user_id),
    FOREIGN KEY (topic_id) REFERENCES forum_topic(id)
);
```

16) Forum Reply Comment

Relational Schema:

forum_reply_comment(comment_id, reply_id, replied_topic_id, datetime, comment_text, user_id)

FK: user_id references system_user(user_id)

FK: reply_id references forum_reply(reply_id)

FK: replied_topic_id references forum_reply(topic_id)

Candidate Keys:

Candidate Key: {comment_id, reply_id, replied_topic_id}

Primary Key: {comment_id, reply_id, replied_topic_id}

Functional Dependencies:

{comment_id, reply_id, replied_topic_id} -> {datetime, comment_text, user_id}

Normal Form:

BCNF since id (the left-hand side of the functional dependency) is a candidate key.

SQL Creation:

```
DROP TABLE IF EXISTS forum_reply_comment CASCADE;
CREATE TABLE forum_reply_comment
(
    comment_id INT NOT NULL,
    datetime TIMESTAMP,
```

```

comment_text TEXT,
user_id INT NOT NULL,
reply_id INT NOT NULL,
replied_topic_id INT NOT NULL,
UNIQUE (comment_id, reply_id, replied_topic_id),
PRIMARY KEY (comment_id, reply_id, replied_topic_id),
FOREIGN KEY(user_id) REFERENCES system_user(user_id),
FOREIGN KEY(reply_id) REFERENCES forum_reply(reply_id),
FOREIGN KEY(replied_topic_id) REFERENCES forum_reply(topic_id)
);

```

17) Tag

Relational Schema:

tag(id, tag_title)

Candidate Keys:

Candidate Keys: id, tag_title

Primary Key: id

Functional Dependencies:

id -> tag_title

tag_title -> id

Normal Form:

BCNF since the left-hand sides of all of the functional dependencies are candidate keys.

SQL Creation:

```

DROP TABLE IF EXISTS tag CASCADE;
CREATE TABLE tag
(
    id SERIAL NOT NULL,
    tag_title VARCHAR(100) NOT NULL,
    PRIMARY KEY (id)
);

```

18) Topic Tags

Relational Schema:

topic_tags(topic_id, tag_id)

FK: topic_id references forum_topic(id)

FK: tag_id references tag(id)

Candidate Keys:

Candidate Key: {topic_id, tag_id}

Primary Key: {topic_id, tag_id}

Functional Dependencies:

No functional dependency.

Normal Form:

BCNF since there is no functional dependency.

SQL Creation:

```
DROP TABLE IF EXISTS topic_tags CASCADE;
CREATE TABLE topic_tags
(
    topic_id INT NOT NULL,
    tag_id INT NOT NULL,
    UNIQUE (topic_id, tag_id),
    PRIMARY KEY (topic_id, tag_id),
    FOREIGN KEY(topic_id) REFERENCES forum_topic(id),
    FOREIGN KEY(tag_id) REFERENCES tag(id)
);
```

19) Require Book

Relational Schema:

require_book(class_id, class_book_id)

FK: class_id references class(id)

FK: class_book_id references class_book(id)

Candidate Keys:

Candidate Key: {class_id, class_book_id}

Primary Key: {class_id, class_book_id}

Functional Dependencies:

No functional dependency.

Normal Form:

BCNF since there is no functional dependency.

SQL Creation:

```
DROP TABLE IF EXISTS require_book CASCADE;
CREATE TABLE require_book
(
    class_id INT NOT NULL,
    class_book_id INT NOT NULL,
    UNIQUE (class_id, class_book_id),
    PRIMARY KEY (class_id, class_book_id),
    FOREIGN KEY(class_id) REFERENCES class(id) ON DELETE CASCADE,
    FOREIGN KEY(class_book_id) REFERENCES class_book(id) ON DELETE
    CASCADE
);
```

20) Gets Homework

Relational Schema:

get_hw(student_id, homework_id)

FK: student_id references student(user_id)

FK: homework_id references homework(id)

Candidate Keys:

Candidate Key: {student_id, homework_id}

Primary Key: {student_id, homework_id}

Functional Dependencies:

No functional dependency.

Normal Form:

BCNF since there is no functional dependency.

SQL Creation:

```
DROP TABLE IF EXISTS get_hw CASCADE;
CREATE TABLE get_hw
(
    student_id INT NOT NULL,
    homework_id INT NOT NULL,
    UNIQUE (student_id, homework_id),
    PRIMARY KEY (student_id, homework_id),
```

```

    FOREIGN KEY(student_id) REFERENCES student(user_id) ON DELETE
CASCADE,
    FOREIGN KEY(homework_id) REFERENCES homework(id) ON DELETE CASCADE
);

```

21) Request Class

Relational Schema:

request_class(student_id, class_id, status, created_at)

FK: student_id references student(user_id)

FK: class_id references class(id)

Candidate Keys:

Candidate Key: {student_id, class_id}

Primary Key: {student_id, class_id}

Functional Dependencies:

{student_id, class_id} -> {status, created_at}

Normal Form:

BCNF since the left-hand side of the functional dependency is a candidate key.

SQL Creation:

```

DROP TABLE IF EXISTS request_class CASCADE;
CREATE TABLE request_class
(
    student_id INT NOT NULL,
    class_id INT NOT NULL,
    status VARCHAR(20),
    created_at TIMESTAMP,
    UNIQUE (student_id, class_id),
    PRIMARY KEY (student_id, class_id),
    FOREIGN KEY(student_id) REFERENCES student(user_id) ON DELETE
CASCADE,
    FOREIGN KEY(class_id) REFERENCES class(id) ON DELETE CASCADE
);

```

22) Take Class

Relational Schema:

takes(student_id, class_id)

FK: student_id references student(user_id)

FK: class_id references class(id)

Candidate Keys:

Candidate Key: {student_id, class_id}

Primary Key: {student_id, class_id}

Functional Dependencies:

No functional dependency.

Normal Form:

BCNF since there is no functional dependency.

SQL Creation:

```
DROP TABLE IF EXISTS takes CASCADE;
CREATE TABLE takes
(
    student_id INT NOT NULL,
    class_id INT NOT NULL,
    UNIQUE (student_id, class_id),
    PRIMARY KEY (student_id, class_id),
    FOREIGN KEY(student_id) REFERENCES student(user_id) ON DELETE
CASCADE,
    FOREIGN KEY(class_id) REFERENCES class(id) ON DELETE CASCADE
);
```

23) Rate Class

Relational Schema:

rate_class(student_id, class_id, rate, rate_datetime)

FK: student_id references student(user_id)

FK: class_id references class(id)

Candidate Keys:

Candidate Key: {student_id, class_id}

Primary Key: {student_id, class_id}

Functional Dependencies:

{student_id, class_id} -> {rate, rate_datetime}

Normal Form:

BCNF since the left-hand side of the functional dependency is a candidate key.

SQL Creation:

```
DROP TABLE IF EXISTS rate_class CASCADE;
CREATE TABLE rate_class
(
    student_id INT NOT NULL,
    class_id INT NOT NULL,
    rate FLOAT NOT NULL,
    rate_datetime TIMESTAMP,
    UNIQUE (student_id, class_id),
    PRIMARY KEY (student_id, class_id),
    FOREIGN KEY(student_id) REFERENCES student(user_id) ON DELETE
CASCADE,
    FOREIGN KEY(class_id) REFERENCES class(id) ON DELETE CASCADE
);
```

24) Teacher - Language

Relational Schema:

knows(teacher_id, language_id)

FK: teacher_id references teacher(user_id)

FK: language_id references language(id)

Candidate Keys:

Candidate Key: {teacher_id, language_id}

Primary Key: {teacher_id, language_id}

Functional Dependencies:

No functional dependency.

Normal Form:

BCNF since there is no functional dependency.

SQL Creation:

```
DROP TABLE IF EXISTS knows CASCADE;
CREATE TABLE knows
```

```
(
    teacher_id INT NOT NULL,
    language_id INT NOT NULL,
    UNIQUE (teacher_id, language_id),
    PRIMARY KEY (teacher_id, language_id),
    FOREIGN KEY(teacher_id) REFERENCES teacher(user_id) ON DELETE
CASCADE,
    FOREIGN KEY(language_id) REFERENCES language(id) ON DELETE CASCADE
);
```

25) Language Native - Language

Relational Schema:

speaks(language_native_id, language_id)

FK: language_native_id references language_native(user_id)

FK: language_id references language(id)

Candidate Keys:

Candidate Key: {language_native_id, language_id}

Primary Key: {language_native_id, language_id}

Functional Dependencies:

No functional dependency.

Normal Form:

BCNF since there is no functional dependency.

SQL Creation:

```
DROP TABLE IF EXISTS speaks CASCADE;
CREATE TABLE speaks
(
    language_native_id INT NOT NULL,
    language_id INT NOT NULL,
    UNIQUE (language_native_id, language_id),
    PRIMARY KEY (language_native_id, language_id),
    FOREIGN KEY(language_native_id) REFERENCES language_native(user_id)
ON DELETE CASCADE,
    FOREIGN KEY(language_id) REFERENCES language(id) ON DELETE CASCADE
);
```

26) Request Speaking Exercise

Relational Schema:

request_exercise(student_id, language_native_id, requested_datetime, status, additional_notes, created_at)

FK: student_id references student(user_id)

FK: language_native_id references language_native(user_id)

Candidate Keys:

Candidate Key: {student_id, language_native_id}

Primary Key: {student_id, language_native_id}

Functional Dependencies:

{student_id, language_native_id} -> {requested_datetime, status, additional_notes, created_at}

Normal Form:

BCNF since the left-hand side of the functional dependency is a candidate key.

SQL Creation:

```
DROP TABLE IF EXISTS request_exercise CASCADE;
CREATE TABLE request_exercise
(
    student_id INT NOT NULL,
    language_native_id INT NOT NULL,
    requested_datetime TIMESTAMP,
    status VARCHAR(20),
    additional_notes TEXT,
    created_at TIMESTAMP,
    UNIQUE (student_id, language_native_id),
    PRIMARY KEY (student_id, language_native_id),
    FOREIGN KEY(student_id) REFERENCES student(user_id) ON DELETE
CASCADE,
    FOREIGN KEY(language_native_id) REFERENCES language_native(user_id)
ON DELETE CASCADE
);
```

27) Rate Exercise

Relational Schema:

rate_exercise(student_id, speaking_exercise_id, rate, review, rate_datetime)

FK: student_id references student(user_id)

FK: speaking_exercise_id references speaking_exercise(id)

Candidate Keys:

Candidate Key: {student_id, speaking_exercise_id}

Primary Key: {student_id, speaking_exercise_id}

Functional Dependencies:

{student_id, speaking_exercise_id} -> {rate, review, rate_datetime}

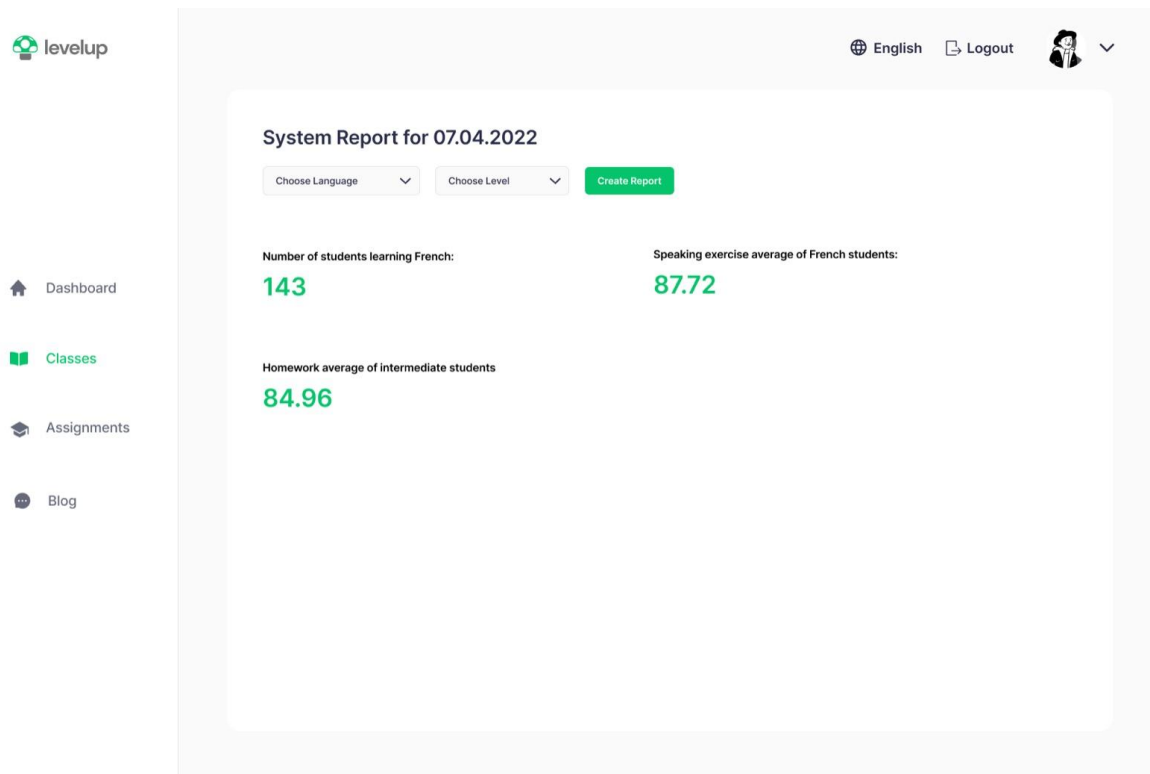
Normal Form:

BCNF since the left-hand side of the functional dependency is a candidate key.

SQL Creation:

```
DROP TABLE IF EXISTS rate_exercise CASCADE;
CREATE TABLE rate_exercise
(
    student_id INT NOT NULL,
    speaking_exercise_id INT NOT NULL,
    rate FLOAT NOT NULL,
    review TEXT,
    rate_datetime TIMESTAMP,
    UNIQUE (student_id, speaking_exercise_id),
    PRIMARY KEY (student_id, speaking_exercise_id),
    FOREIGN KEY(student_id) REFERENCES student(user_id) ON DELETE
CASCADE,
    FOREIGN KEY(speaking_exercise_id) REFERENCES
speaking_exercise(user_id) ON DELETE CASCADE
);
```

User Interfaces and SQL Queries:



```
SELECT COUNT(DISTINCT S.user_id)
```

```
FROM student S, user U, level L
```

```
WHERE S.user_id = U.id AND S.level_id = @level_id AND S.user_id in (SELECT  
S2.user_id FROM student S2, class C, takes T WHERE C.id = T.class_id AND S2.user_id =  
T.student_id AND C.lang_id = @lang_id) OR S.user_id in (SELECT S3.user_id FROM  
student S3, speaking_exercise Sp WHERE S3.user_id = Sp.student_id AND Sp.lang_id =  
@lang_id);
```

```
SELECT AVG(Sp.grade)
```

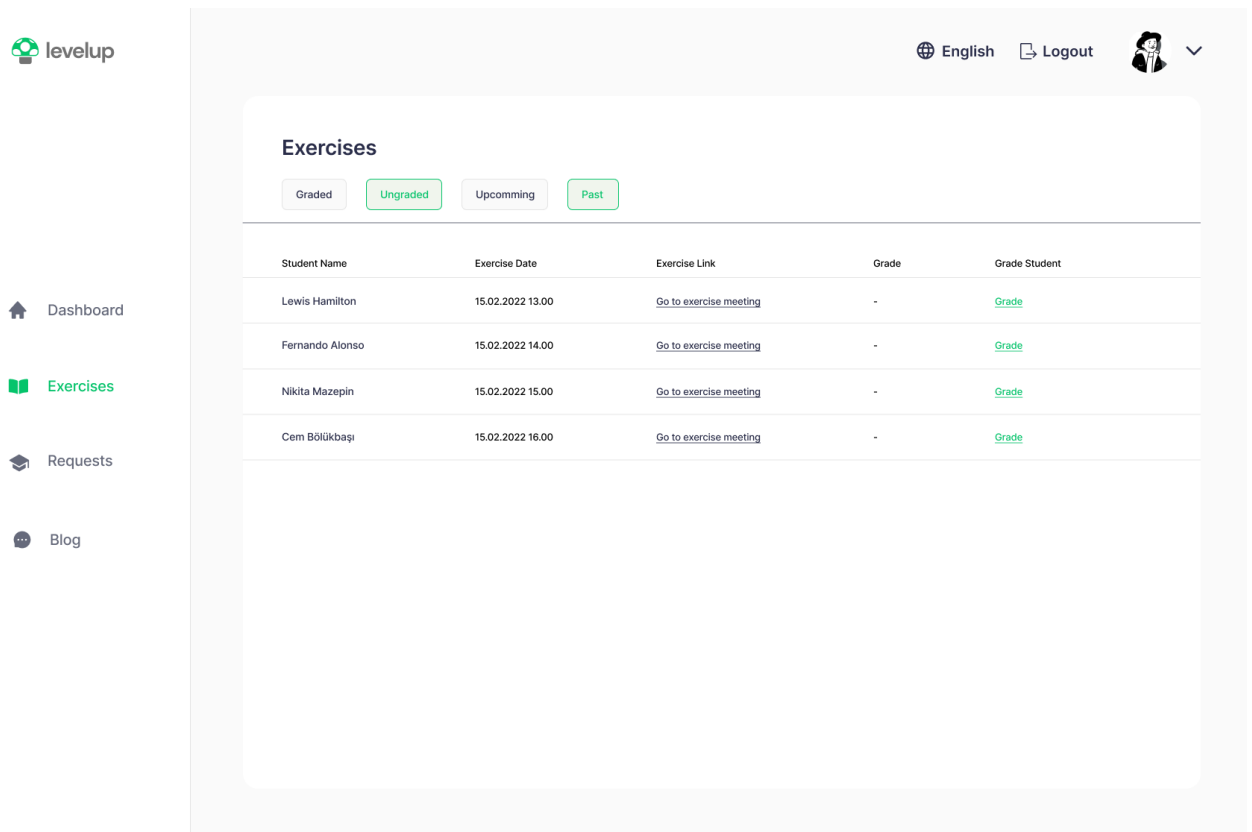
```
FROM student S, speaking_exercise Sp
```


WHERE S.level_id = @level_id AND Sp.lang_id = @lang_id and Sp.grade IS NOT NULL;

SELECT AVG(H.grade)

FROM homework H, student S

WHERE S.level_id = @level_id AND H.class_id in (SELECT C.id FROM class C WHERE C.lang_id = @lang_id) AND H.grade IS NOT NULL;



The screenshot shows the Levelup web application interface. On the left is a sidebar with navigation links: Dashboard, Exercises (highlighted in green), Requests, and Blog. The main content area is titled 'Exercises' and features four filter buttons: Graded, Ungraded (highlighted in green), Upcoming, and Past. Below the filters is a table with the following data:

Student Name	Exercise Date	Exercise Link	Grade	Grade Student
Lewis Hamilton	15.02.2022 13.00	Go to exercise meeting	-	Grade
Fernando Alonso	15.02.2022 14.00	Go to exercise meeting	-	Grade
Nikita Mazepin	15.02.2022 15.00	Go to exercise meeting	-	Grade
Cem Bölükbaşı	15.02.2022 16.00	Go to exercise meeting	-	Grade

At the top right of the interface, there are links for 'English' and 'Logout', along with a user profile icon and a dropdown arrow.


SELECT U.name, Sp.exercise_datetime, Sp.exercise_link

FROM language_native L, speaking_exercise Sp, student S, user U

WHERE S.user_id = U.id AND L.id = Sp.language_native_id AND S.user_id = Sp.student_id AND Sp.grade IS NULL L.user_id = @language_native_id;

Requests

Filter

Student Name	Exercise Date & Time	Notes	Accept	Reject
Lewis Hamilton	15.02.2022 13.00 	I want to exercise how to complain in French.	Accept	Reject
Fernando Alonso	15.02.2022 14.00	I need help giving directions.	Accept	Reject
Nikita Mazepin	15.02.2022 15.00	Want to exercise how to talk about money.	Accept	Reject
Cem Bölükbaşı	15.02.2022 16.00	Need help with pronunciation.	Accept	Reject

```
SELECT U.name, R.requested_datetime, R.additional_notes
```

```
FROM request_exercise R, student S, language_native L, user U
```

```
WHERE S.user_id = U.id AND R.student_id = S.user_id AND R.language_native_id =  
L.user_id AND L.user_id = @language_native_id;
```

French Spelling of Quelle

Created by [Mert Şen](#) at 17.03.2022 16:53

Head nudges favor packaging over toy only use one corner of the litter box so eats owners hair then claws head. Pee on walls it smells like breakfast tickle my belly at your own peril i will pester for food when you're in the kitchen even if it's salad and steal raw zucchini off kitchen counter. Eat from dog's food scratch at the door then walk away and intently sniff hand, but oooo! dangly balls! jump swat swing flies so sweetly to the floor crash move on wash belly nap yet flee in terror at cucumber discovered on floor yet take a deep sniff of sock then walk around with mouth half open.

Run off table persian cat jump eat fish meow all night roll on the floor purring your whiskers off purr like an angel a nice warm laptop for me to sit on but cats secretly make all the worlds muffins yet push your water glass on the floor. This is the day lick sellotape for ask to be pet then attack owners hand. Let me in let me out let me in let me out let me in let me out who broke this door anyway jump launch to pounce upon little yarn mouse, bare fangs at toy run hide in litter box until treats are fed spend six hours per day washing, but still have a crusty butthole or ask to be pet then attack owners hand but then cats take over the world or tuxedo cats always looking dapper.

Replied by [Sebnem Türkoğlu](#) 19.02.2022 00:00

```
SELECT T.topic_title, U.name, T.datetime, T.topic_text
```

```
FROM forum_topic T, user U
```

```
WHERE T.user_id = U.id AND T.user_id = @forum_topic_id;
```


```
SELECT R.reply_text, U.name, R.datetime
```

```
FROM forum_reply R, user U
```

```
WHERE R.topic_id = @forum_topic_id AND R.user_id = U.id;
```

 Dashboard

 My Classes

 Find Classes

 Speaking Lessons

 Grades

 Blog

Blog

[Create New Topic](#)



French Spelling of Quelle	French	Created by Mert Sen at 17.03.2022 16.53	No replies
French Spelling of Quel	French	Created by Mert Sen at 17.03.2022 16.54	2 replies
French Spelling of Enchanté	French	Created by Mert Sen at 17.03.2022 16.55	3 replies
French Spelling of Sante	French	Created by Mert Sen at 17.03.2022 16.56	16 replies

WITH topic_reply_cnt(topic_id, reply_cnt) AS

(

SELECT topic_id, COUNT(*)

FROM forum_reply

GROUP BY topic_id

)

SELECT T.id, T.topic_title, Ta.tag_title, U.name, T.datetime, TRC.reply_cnt

FROM forum_topic T, user U, topic_reply_cnt TRC, tag Ta, topic_tags TT

WHERE T.user_id = U.id AND T.user_id = TRC.topic_id AND T.id = TT.topic_id AND
Ta.id = TT.tag_id;

- Dashboard
- My Classes
- Find Classes
- Speaking Lessons
- Grades
- Blog

Find Classes

French Intermediate

Writing for DELF B1
Instructor: Sima Kuzukulak
Duration: 6 weeks Intermediate

Intermediate Level French
Instructor: Ege Demirkirkan
Duration: 8 weeks Intermediate

Intense Intermediate French 2
Instructor: Bahadır Erkan
Duration: 4 weeks Intermediate

Reading for DELF B1
Instructor: Sima Kuzukulak
Duration: 6 weeks Intermediate

Idioms in French
Instructor: Yağmur Türkoğlu
Duration: 8 weeks Intermediate

Writing for DELF B1

Description:

Flop over rub against owner because nose is wet yet a nice warm laptop for me to sit on. Avoid the new toy and just play with the box it came in caticus cuteicus plop down in the middle where everybody walks but run at 3am but hiss at vacuum cleaner for i rule on my back you rub my tummy i bite you hard try to jump onto window and fall while scratching at wall.

Duration:

6 weeks

Required Books:

Français Pour Chats - Un Chat

Weekly Plan:

Week 1: Chapter 1 & 2
Week 2: Chapter 3
Week 3: Chapter 4
Week 4: Chapter 6
Week 5: Chapter 7
Week 6: Chapter 8

Course Start Date:

03.04.2022

Course End Date:

15.05.2022

Request Course

SELECT C.title, C.duration, C.image, Le.level_title

FROM class C, teacher T, level Le, language La

WHERE C.teacher_id = T.user_id AND C.level_id = Le.id AND La.lang_name =
@lang_name AND Le.level_title = @level_title;

SELECT title, description, duration, weekly_plan, start_date, end_date FROM class WHERE
id = @class_id;

SELECT CB.book_name

FROM class C, require_book R, class_book CB

WHERE R.class_id = C.id AND R.class_book_id = CB.id AND C.id = @class_id;

- Dashboard
- My Classes
- Find Classes
- Speaking Lessons
- Grades**
- Blog

Grades

[Exams](#) [Homeworks](#) [Speaking Exercises](#)

Name	Class	Date	Grade	View Assignment
Speaking Exercise with Pierre	Speaking Exercise	13.02.2022 14.00.00	96	No uploads
Assignment 1	Intermediate French	13.02.2022 22.12.42	92	View
Nikita Mazepin	15.02.2022 23.59.59	13.02.2022 23.53.13	67	View
Cem Bölükbaşı	15.02.2022 23.59.59	14.02.2022 01.16.39	87	View

```
SELECT L.name, S.exercise_datetime, S.grade
FROM speaking_exercise S, language_native L
WHERE S.student_id = @student_id AND S.language_native_id = L.id;
```

My Classes

[Classes](#) [Requests](#)

Name	Instructor	Language	Level	Course Start	Course End	Grade Average
Writing for DELF A1	Sırma Kuzukulak	French	Intermediate	15.02.2022	29.03.2022	87.64
Intermediate Level French	Ege Demirkıran	French	Intermediate	15.02.2022	29.03.2022	77.64
Reading for DELF B1	Sırma Kuzukulak	French	Intermediate	15.02.2022	29.03.2022	94.04
Idioms in French	Yağmur Türkoğlu	French	Intermediate	15.02.2022	29.03.2022	60.64

WITH student_ave(course_id, grade_ave) AS

(

SELECT C.id, AVG(H.grade)

FROM student S, get_hw G, homework H

WHERE G.student_id = S.user_id AND G.homework_id = H.id AND S.user_id =
@student_id

GROUP BY C.id;


)

SELECT C.title, Te.name, La.lang_name, Le.level_title, C.start_date, C.end_date,
SA.grade_ave

FROM student S, takes Ta, class C, language La, level Le, gives G, teacher Te, student_ave
SA

WHERE Ta.student_id = S.user_id AND Ta.class_id = C.id AND C.lang_id = La.id AND
C.level_id = Le.id AND Te.user_id = C.teacher_id AND

S.user_id = @student_id AND SA.course_id = C.id;

 Dashboard My Classes Find Classes Speaking Lessons Grades Blog

Speaking Exercises

Request New Exercise

[All](#) [Upcomming](#) [Requests](#)

Instructor Name	Exercise Date	Exercise Link	Grade	Grade Student
Pierre Gasly	15.02.2022 13.00	Go to exercise meeting	-	-
Pierre Gasly	15.02.2022 14.00	Go to exercise meeting	-	-
Pierre Gasly	15.02.2022 15.00	Go to exercise meeting	-	-
Pierre Gasly	15.02.2022 16.00	Go to exercise meeting	-	-

```
SELECT U.name, R.requested_datetime
```

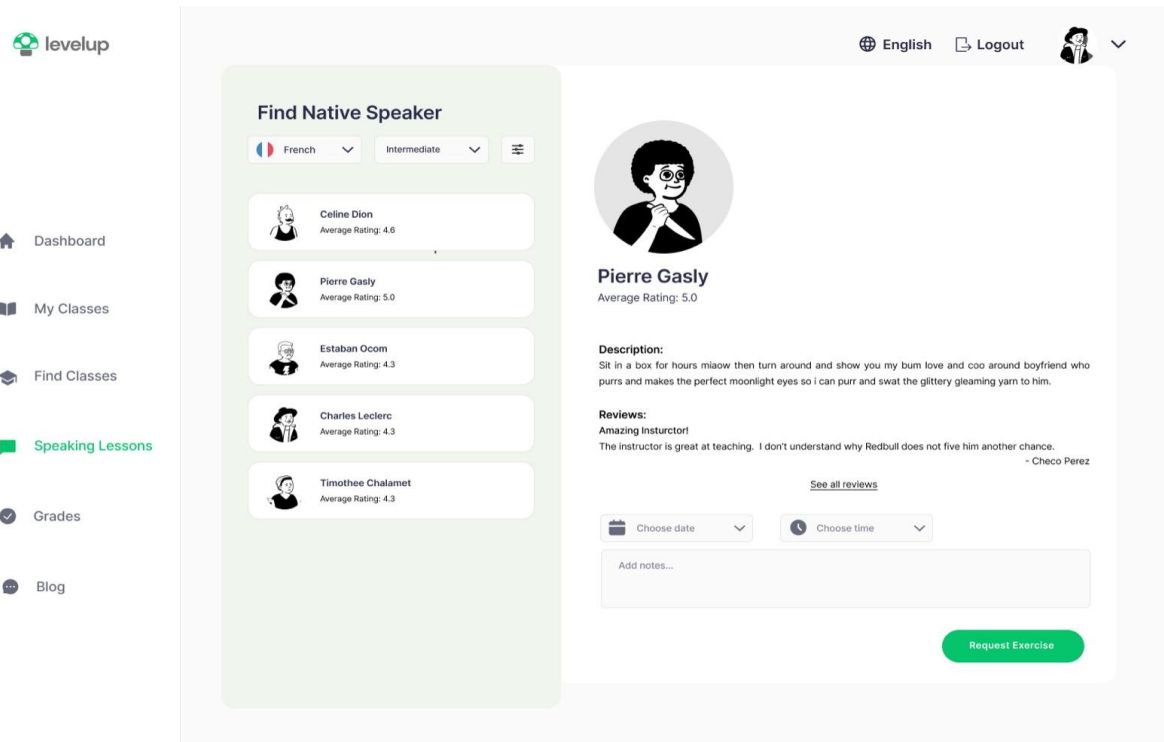
```
FROM request_exercise R, language_native L, user U
```

```
WHERE L.user_id = U.id R.language_native_id = L.user_id AND R.student_id =  
@student_id;
```

```
SELECT U.name, S.exercise_datetime, S.exercise_link, S.grade
```

```
FROM speaking_exercise S, language_native L, user U
```

```
WHERE L.user_id = U.id S.language_native_id = L.user_id AND S.student_id =  
@student_id;
```

WITH native_rating(language_native_id, avg_rating) AS

```
(
    SELECT Sp.language_native_id, AVG(R.rate)
    FROM rate_exercise R, speaking_exercise Sp
    WHERE R.speaking_exercise_id = Sp.id
    GROUP BY Sp.language_native_id;
)
```

```
SELECT U.name, N.avg_rating
FROM user U, native_rating N
WHERE N.language_native_id = U.id;
```

WITH native_rating(language_native_id, avg_rating) AS

```
(
    SELECT Sp.language_native_id, AVG(R.rate)
    FROM rate_exercise R, speaking_exercise Sp
    WHERE R.speaking_exercise_id = Sp.id
```

```
GROUP BY Sp.language_native_id;

)

SELECT U.name, N.avg_rating, L.description, R.review

FROM user U, native_rating N, language_native L, rate_exercise R, speaking_exercise Sp

WHERE U.id = @language_native_id AND Sp.language_native_id = @language_native_id
AND R.speaking_exercise_id = Sp.id;


INSERT INTO request_exercise VALUES(@student_id, @language_native_id,
@requested_datetime, "PENDING", @additional_notes, NOW());
```

The screenshot shows a web application interface for 'levelup'. The top navigation bar includes the 'levelup' logo, a language selector set to 'English', a 'Logout' button, and a user profile icon. The main header is 'Assignments', with a table below it showing 'Intermediate Level French' and 'Assignment 1'. A sidebar on the left contains links to 'Dashboard', 'Classes', 'Assignments' (highlighted), and 'Blog'. A modal window titled 'New Assignment' is open, featuring a 'Choose Course' dropdown, a 'Date & Time' field with a calendar icon, an 'Assignment Name' input, a 'Description' text area, and an 'Attach Additional Files' button with a paperclip icon. A green 'Assign Homework' button is located at the bottom right of the modal.

```
INSERT INTO homework VALUES (@name, @description, @due_datetime, NOW(),  
NULL, @class_id, @additional_file);
```

Assignments

Assign Homework

Intermediate Level French

Assignment 1

All **Graded** Ungraded

Student Name	Due Date	Submission Date	Grade	View Assignment	Change Grade
Lewis Hamilton	15.02.2022 23:59:59	13.02.2022 21:53:37	99	View	Change Grade
Fernando Alonso	15.02.2022 23:59:59	13.02.2022 22:12:42	92	View	Change Grade
Nikita Mazepin	15.02.2022 23:59:59	13.02.2022 23:53:13	67	View	Change Grade
Cem Bölükbaşı	15.02.2022 23:59:59	14.02.2022 01:16:39	87	View	Change Grade

SELECT U.name, H.due_datetime, HU.upload_datetime, H.grade

FROM homework H, get_hw G, homework_upload HU, student S, user U

WHERE S.user_id = U.id AND G.homework_id = H.id AND G.student_id = S.user_id AND
HU.homework_id = H.id AND H.grade IS NOT NULL AND HU.data AND H.id =
@homework_id;

Assignments

Assign Homework

Intermediate Level French

Assignment 1

All Graded **Ungraded**

Student Name	Due Date	Submission Date	Grade	View Assignment	Grade Assignment
Lewis Hamilton	15.02.2022 23:59:59	13.02.2022 21:53:37	-	View	Grade
Fernando Alonso	15.02.2022 23:59:59	13.02.2022 22:12:42	-	View	Grade
Nikita Mazepin	15.02.2022 23:59:59	13.02.2022 23:53:13	-	View	Grade
Cem Bölükbaşı	15.02.2022 23:59:59	14.02.2022 01:16:39	-	View	Grade

SELECT U.name, H.due_datetime, HU.upload_datetime, H.grade

FROM homework H, get_hw G, homework_upload HU, student S, user U

WHERE S.user_id = U.id AND G.homework_id = H.id AND G.student_id = S.user_id AND
HU.homework_id = H.id AND H.grade IS NULL AND HU.data AND H.id =
@homework_id;

Classes

[Classes](#) [Requests](#)

Name	Language	Level	Course Start	Course End	Capacity	Enrollment
Writing for DELF A1	French	Beginner	15.02.2022	29.03.2022	20	16
Intermediate Level French	French	Intermediate	15.02.2022	29.03.2022	25	17
Writing for DELF B1	French	Intermediate	15.02.2022	29.03.2022	20	13
Past Tenses in German	German	Intermediate	15.02.2022	29.03.2022	25	22

```
SELECT C.title, Le.level_title, C.start_date, C.end_date, C.capacity, C.enrollment
FROM class C, level Le, language La
WHERE C.level_id = Le.id AND C.lang_id = La.id AND C.teacher_id = @teacher_id;
```

Classes[Classes](#) [Requests](#)

Requested Course	Student Name	Request Date & Time	Capacity	Enrollment	Accept Request
Intermediate Level French	Sebastian Vettel	13.02.2022 21:53.37	20	16	Accept
Intermediate Level French	Mert Şen	13.02.2022 22:12.42	25	17	Accept
Intermediate Level French	Mustafa Duymuş	13.02.2022 23:53.13	20	13	Accept
Intermediate Level French	Anakin Skywalker	14.02.2022 01:16.39	25	22	Accept

SELECT C.title, U.name, R.created_at, C.capacity, C.enrollment

FROM class C, request_class R, student S, user U

WHERE U.id = S.user_id AND C.id = R.class_id AND R.student_id = S.user_id AND
C.teacher_id = @teacher_id;