ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

Катедра „Компютърни системи”

КУРСОВ ПРОЕКТ

ПО БАЗИ ОТ ДАННИ

Студент: .....

ФАК. № ..... Група:.....

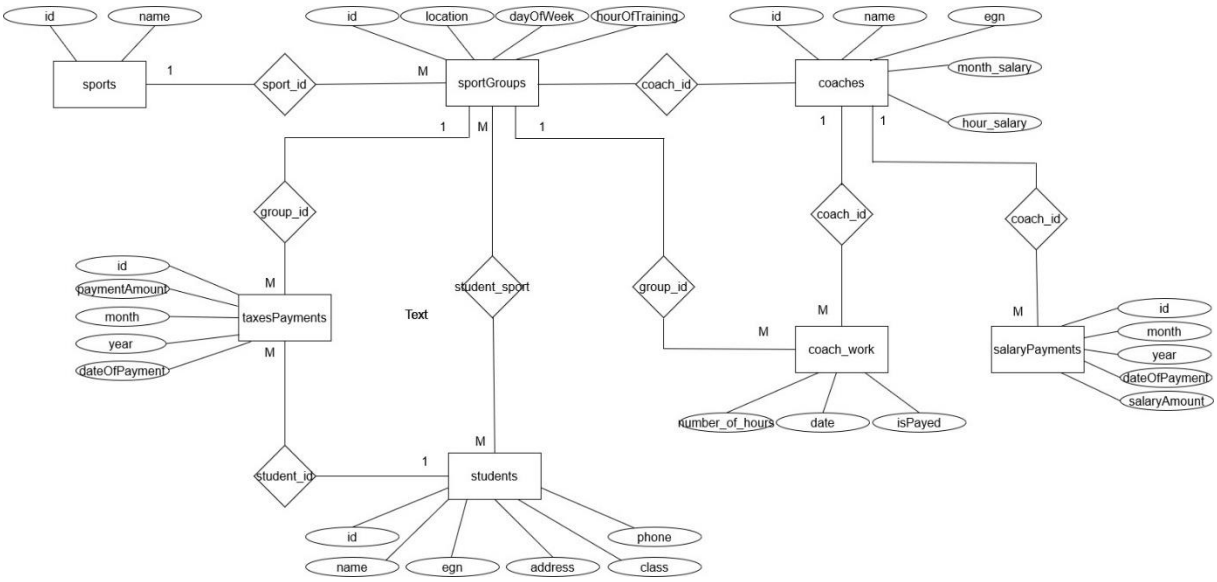
Тема №…
По даден проект училище трябва да организира извънучилищни спортни клубове. Спортните клубове могат да бъдат първоначално два типа– по футбол и по волейбол, а на следващ етап ще се добавят още. За всеки клуб може да има много наброй групи, които тренират в различни дни и в различни часове. Учениците могат да се присъединят към който и да е от двата типа клуба едновременно като се запишат в определена група в определен ден от седмицата и в определен час. За всяка от клубните групи има по един треньор, който тренира учениците. Всеки ученик може да се идентифицира с уникален номер – id. Допълнете таблиците с необходима информация по ваш избор.

1. Да се проектира база от данни и да се представи ER диаграма със съответни CREATE TABLE заявки за средата MySQL.
2. Напишете заявка, в която демонстрирате SELECT с ограничаващо условие по избор.
3. Напишете заявка, в която използвате агрегатна функция и GROUP BY по ваш избор.
4. Напишете заявка, в която демонстрирате INNER JOIN по ваш избор.
5. Напишете заявка, в която демонстрирате OUTER JOIN по ваш избор.
6. Напишете заявка, в която демонстрирате вложен SELECT по ваш избор.
7. Напишете заявка, в която демонстрирате едновременно JOIN и агрегатна функция.
8. Създайте тригер по ваш избор.
9. Създайте процедура, в която демонстрирате използване на курсор.

Вашата работа трябва да включва: задание, ER-диаграма, CREATE TABLE заявки, всички останали заявки, решения на задачите от 2 до 9 и резултатите от тях.

1. Да се проектира база от данни и да се представи ER диаграма със съответни CREATE TABLE заявки за средата MySQL:

Основните обекти, за които трябва да съхраняваме информация, според заданието са: SportGroups, Students, Coaches и Sports. Допълнително ще създадем още няколко таблици. Първата ще отразява плащанията на таксите за всеки ученик. В нея ще се съдържа информация за това кой плаща, за коя група, за кой месец, за коя година, сума и дата, на която е извършено плащането. В друга таблица ще съхраняваме месечните заплати на треньорите - кой каква сума е получил, за кой месец, година и дата на превод. В таблицата coaches ще добавим колона, която показва колко е фиксираната месечна заплата на всеки треньор, както и колона, в която ще записваме колко е хонорара на час за дадения треньор. В следващата таблица, която ще създадем - coach_work, ще отразяваме часовете които прави всеки треньор, за да могат да се изплатят и хонорарите за конкретния месец. Накрая ще създадем лог таблица, в която да се отразяват всички промени на salarypayments. За проектирането на базата ще използваме модела ER-диаграма (Entity Relationship Diagram):



Заявките, с които създаваме базата данни и таблиците са:

```
CREATE DATABASE school_sport_clubs;

CREATE TABLE school_sport_clubs.sports(
    id INT AUTO_INCREMENT PRIMARY KEY ,
    name VARCHAR(255) NOT NULL
)Engine = InnoDB;

CREATE TABLE school_sport_clubs.coaches(
```

```
        id INT AUTO_INCREMENT PRIMARY KEY ,
        name VARCHAR(255) NOT NULL ,
        egn VARCHAR(10) NOT NULL UNIQUE CONSTRAINT EGN
CHECK(CHAR_LENGTH(egn) = 10),
        month_salary DECIMAL ,
        hour_salary DECIMAL
    )Engine = InnoDB;

    CREATE TABLE school_sport_clubs.students(
        id INT AUTO_INCREMENT PRIMARY KEY ,
        name VARCHAR(255) NOT NULL ,
        egn VARCHAR(10) NOT NULL UNIQUE ,
        address VARCHAR(255) NOT NULL ,
        phone VARCHAR(20) NULL DEFAULT NULL ,
        class VARCHAR(10) NULL DEFAULT NULL
    )Engine = InnoDB;

    CREATE TABLE school_sport_clubs.sportGroups(
        id INT AUTO_INCREMENT PRIMARY KEY ,
        location VARCHAR(255) NOT NULL ,
        dayOfWeek
ENUM('Monday','Tuesday','Wednesday','Thursday','Friday','Saturday','Sunda
y') ,
        hourOfTraining TIME NOT NULL ,
        sport_id INT NOT NULL ,
        coach_id INT NOT NULL ,
        UNIQUE KEY(location,dayOfWeek,hourOfTraining)  ,
        CONSTRAINT FOREIGN KEY(sport_id)
            REFERENCES sports(id) ,
        CONSTRAINT FOREIGN KEY (coach_id)
            REFERENCES coaches(id)
    )Engine = InnoDB;

    CREATE TABLE school_sport_clubs.student_sport(
        student_id INT NOT NULL ,
        sportGroup_id INT NOT NULL ,
        CONSTRAINT FOREIGN KEY (student_id)
            REFERENCES students(id) ,
        CONSTRAINT FOREIGN KEY (sportGroup_id)
            REFERENCES sportGroups(id) ,
        PRIMARY KEY(student_id,sportGroup_id)
    )Engine = InnoDB;

    CREATE TABLE taxesPayments(
        id INT AUTO_INCREMENT PRIMARY KEY,
        student_id INT NOT NULL,
        group_id INT NOT NULL,
        paymentAmount DOUBLE NOT NULL,
        month TINYINT,
        year YEAR,
        dateOfPayment DATETIME NOT NULL ,
        CONSTRAINT FOREIGN KEY (student_id)
            REFERENCES students(id),
        CONSTRAINT FOREIGN KEY (group_id)
            REFERENCES sportgroups(id)
    )Engine = InnoDB;

    CREATE TABLE salaryPayments(
        id INT AUTO_INCREMENT PRIMARY KEY,
        coach_id INT NOT NULL,
        month TINYINT,
```

```
        year YEAR,
        salaryAmount DOUBLE CONSTRAINT salaryCantBeNegative
CHECK(salaryAmount >= 0),
        dateOfPayment DATETIME not null,
        CONSTRAINT FOREIGN KEY (coach_id)
            REFERENCES coaches(id),
        UNIQUE KEY(`coach_id`,`month`,`year`)
    )Engine = InnoDB;

    CREATE TABLE coach_work(
        id INT AUTO_INCREMENT PRIMARY KEY,
        coach_id INT NOT NULL,
        group_id INT NOT NULL,
        number_of_hours INT NOT NULL DEFAULT 1,
        date DATETIME NOT NULL,
        isPayed BOOLEAN NOT NULL DEFAULT 0,
        CONSTRAINT FOREIGN KEY (coach_id) REFERENCES coaches(id),
        CONSTRAINT FOREIGN KEY (group_id) REFERENCES sportgroups(id)
    )Engine = InnoDB;

    CREATE TABLE salarypayments_log(
    id INT AUTO_INCREMENT PRIMARY KEY,
    operation ENUM('INSERT','UPDATE','DELETE') NOT NULL,
    old_coach_id INT,
    new_coach_id INT,
    old_month INT,
    new_month INT,
    old_year INT,
    new_year INT,
    old_salaryAmount DECIMAL,
    new_salaryAmount DECIMAL,
    old_dateOfPayment DATETIME,
    new_dateOfPayment DATETIME,
    dateOfLog DATETIME
    )Engine = InnoDB;
```

Добавяме и тестови данни в таблиците:

```
INSERT INTO sports
VALUES  (NULL, 'Football') ,
        (NULL, 'Volleyball'),
        (NULL, 'Tennis');

INSERT INTO coaches  (name, egn)
VALUES  ('Ivan Todorov Petkov', '7509041245') ,
        ('georgi Ivanov Todorov', '8010091245') ,
        ('Ilian Todorov Georgiev', '8407106352') ,
        ('Petar Slavkov Yordanov', '7010102045') ,
        ('Todor Ivanov Ivanov', '8302160980') ,
        ('Slavi Petkov Petkov', '7106041278');

INSERT INTO students (name, egn, address, phone, class)
VALUES     ('Iliyan Ivanov', '9401150045', 'Sofia-Mladost 1',
'0893452120', '10') ,
        ('Ivan Iliev Georgiev', '9510104512', 'Sofia-Liylin',
'0894123456', '11') ,
```

```
        ('Elena Petrova Petrova', '9505052154', 'Sofia-Mladost 3',
'0897852412', '11') ,
        ('Ivan Iliev Iliev', '9510104542', 'Sofia-Mladost 3',
'0894123457', '11') ,
        ('Maria Hristova Dimova', '9510104547', 'Sofia-Mladost 4',
'0894123442', '11') ,
        ('Antoaneta Ivanova Georgieva', '9411104547', 'Sofia-Krasno
selo', '0874526235', '10');

INSERT INTO sportGroups
VALUES  (NULL, 'Sofia-Mladost 1', 'Monday', '08:00:00', 1, 1 ) ,
        (NULL, 'Sofia-Mladost 1', 'Monday', '09:30:00', 1, 2 ) ,
        (NULL, 'Sofia-Liylin 7', 'Sunday', '08:00:00', 2, 1) ,
        (NULL, 'Sofia-Liylin 7', 'Sunday', '09:30:00', 2, 2) ,
        (NULL, 'Plovdiv', 'Monday', '12:00:00', '1', '1');

INSERT INTO student_sport
VALUES  (1, 1),
        (2, 1),
        (3, 1),
        (4, 2),
        (5, 2),
        (6, 2),
        (1, 3),
        (2, 3),
        (3, 3);

INSERT INTO `school_sport_clubs`.`taxespayments`
VALUES  (NULL, '1', '1', '200', '1', 2022, now()),
        (NULL, '1', '1', '200', '2', 2022, now()),
        (NULL, '1', '1', '200', '3', 2022, now()),
        (NULL, '1', '1', '200', '4', 2022, now()),
        (NULL, '1', '1', '200', '5', 2022, now()),
        (NULL, '1', '1', '200', '6', 2022, now()),
        (NULL, '1', '1', '200', '7', 2022, now()),
        (NULL, '1', '1', '200', '8', 2022, now()),
        (NULL, '1', '1', '200', '9', 2022, now()),
        (NULL, '1', '1', '200', '10', 2022, now()),
        (NULL, '1', '1', '200', '11', 2022, now()),
        (NULL, '1', '1', '200', '12', 2022, now()),
        (NULL, '2', '1', '250', '1', 2022, now()),
        (NULL, '2', '1', '250', '2', 2022, now()),
        (NULL, '2', '1', '250', '3', 2022, now()),
        (NULL, '2', '1', '250', '4', 2022, now()),
        (NULL, '2', '1', '250', '5', 2022, now()),
        (NULL, '2', '1', '250', '6', 2022, now()),
        (NULL, '2', '1', '250', '7', 2022, now()),
        (NULL, '2', '1', '250', '8', 2022, now()),
        (NULL, '2', '1', '250', '9', 2022, now()),
        (NULL, '2', '1', '250', '10', 2022, now()),
        (NULL, '2', '1', '250', '11', 2022, now()),
        (NULL, '2', '1', '250', '12', 2022, now()),
        (NULL, '3', '1', '250', '1', 2022, now()),
        (NULL, '3', '1', '250', '2', 2022, now()),
        (NULL, '3', '1', '250', '3', 2022, now()),
        (NULL, '3', '1', '250', '4', 2022, now()),
        (NULL, '3', '1', '250', '5', 2022, now()),
        (NULL, '3', '1', '250', '6', 2022, now()),
        (NULL, '3', '1', '250', '7', 2022, now()),
        (NULL, '3', '1', '250', '8', 2022, now()),
        (NULL, '3', '1', '250', '9', 2022, now()),
```

```
        (NULL, '3', '1', '250', '10', 2022, now()),
        (NULL, '3', '1', '250', '11', 2022, now()),
        (NULL, '3', '1', '250', '12', 2022, now()),
        (NULL, '1', '2', '200', '1', 2022, now()),
        (NULL, '1', '2', '200', '2', 2022, now()),
        (NULL, '1', '2', '200', '3', 2022, now()),
        (NULL, '1', '2', '200', '4', 2022, now()),
        (NULL, '1', '2', '200', '5', 2022, now()),
        (NULL, '1', '2', '200', '6', 2022, now()),
        (NULL, '1', '2', '200', '7', 2022, now()),
        (NULL, '1', '2', '200', '8', 2022, now()),
        (NULL, '1', '2', '200', '9', 2022, now()),
        (NULL, '1', '2', '200', '10', 2022, now()),
        (NULL, '1', '2', '200', '11', 2022, now()),
        (NULL, '1', '2', '200', '12', 2022, now()),
        (NULL, '4', '2', '200', '1', 2022, now()),
        (NULL, '4', '2', '200', '2', 2022, now()),
        (NULL, '4', '2', '200', '3', 2022, now()),
        (NULL, '4', '2', '200', '4', 2022, now()),
        (NULL, '4', '2', '200', '5', 2022, now()),
        (NULL, '4', '2', '200', '6', 2022, now()),
        (NULL, '4', '2', '200', '7', 2022, now()),
        (NULL, '4', '2', '200', '8', 2022, now()),
        (NULL, '4', '2', '200', '9', 2022, now()),
        (NULL, '4', '2', '200', '10', 2022, now()),
        (NULL, '4', '2', '200', '11', 2022, now()),
        (NULL, '4', '2', '200', '12', 2022, now()),
        /**2021**/
        (NULL, '1', '1', '200', '1', 2021, now()),
        (NULL, '1', '1', '200', '2', 2021, now()),
        (NULL, '1', '1', '200', '3', 2021, now()),
        (NULL, '1', '1', '200', '4', 2021, now()),
        (NULL, '1', '1', '200', '5', 2021, now()),
        (NULL, '1', '1', '200', '6', 2021, now()),
        (NULL, '1', '1', '200', '7', 2021, now()),
        (NULL, '1', '1', '200', '8', 2021, now()),
        (NULL, '1', '1', '200', '9', 2021, now()),
        (NULL, '1', '1', '200', '10', 2021, now()),
        (NULL, '1', '1', '200', '11', 2021, now()),
        (NULL, '1', '1', '200', '12', 2021, now()),
        (NULL, '2', '1', '250', '1', 2021, now()),
        (NULL, '2', '1', '250', '2', 2021, now()),
        (NULL, '2', '1', '250', '3', 2021, now()),
        (NULL, '2', '1', '250', '4', 2021, now()),
        (NULL, '2', '1', '250', '5', 2021, now()),
        (NULL, '2', '1', '250', '6', 2021, now()),
        (NULL, '2', '1', '250', '7', 2021, now()),
        (NULL, '2', '1', '250', '8', 2021, now()),
        (NULL, '2', '1', '250', '9', 2021, now()),
        (NULL, '2', '1', '250', '10', 2021, now()),
        (NULL, '2', '1', '250', '11', 2021, now()),
        (NULL, '2', '1', '250', '12', 2021, now()),
        (NULL, '3', '1', '250', '1', 2021, now()),
        (NULL, '3', '1', '250', '2', 2021, now()),
        (NULL, '3', '1', '250', '3', 2021, now()),
        (NULL, '3', '1', '250', '4', 2021, now()),
        (NULL, '3', '1', '250', '5', 2021, now()),
        (NULL, '3', '1', '250', '6', 2021, now()),
        (NULL, '3', '1', '250', '7', 2021, now()),
        (NULL, '3', '1', '250', '8', 2021, now()),
        (NULL, '3', '1', '250', '9', 2021, now()),
```

```
        (NULL, '3', '1', '250', '10', 2021, now()),
        (NULL, '3', '1', '250', '11', 2021, now()),
        (NULL, '3', '1', '250', '12', 2021, now()),
        (NULL, '1', '2', '200', '1', 2021, now()),
        (NULL, '1', '2', '200', '2', 2021, now()),
        (NULL, '1', '2', '200', '3', 2021, now()),
        (NULL, '1', '2', '200', '4', 2021, now()),
        (NULL, '1', '2', '200', '5', 2021, now()),
        (NULL, '1', '2', '200', '6', 2021, now()),
        (NULL, '1', '2', '200', '7', 2021, now()),
        (NULL, '1', '2', '200', '8', 2021, now()),
        (NULL, '1', '2', '200', '9', 2021, now()),
        (NULL, '1', '2', '200', '10', 2021, now()),
        (NULL, '1', '2', '200', '11', 2021, now()),
        (NULL, '1', '2', '200', '12', 2021, now()),
        (NULL, '4', '2', '200', '1', 2021, now()),
        (NULL, '4', '2', '200', '2', 2021, now()),
        (NULL, '4', '2', '200', '3', 2021, now()),
        (NULL, '4', '2', '200', '4', 2021, now()),
        (NULL, '4', '2', '200', '5', 2021, now()),
        (NULL, '4', '2', '200', '6', 2021, now()),
        (NULL, '4', '2', '200', '7', 2021, now()),
        (NULL, '4', '2', '200', '8', 2021, now()),
        (NULL, '4', '2', '200', '9', 2021, now()),
        (NULL, '4', '2', '200', '10', 2021, now()),
        (NULL, '4', '2', '200', '11', 2021, now()),
        (NULL, '4', '2', '200', '12', 2021, now()),
        /**2020**/
        (NULL, '1', '1', '200', '1', 2020, now()),
        (NULL, '1', '1', '200', '2', 2020, now()),
        (NULL, '1', '1', '200', '3', 2020, now()),
        (NULL, '2', '1', '250', '1', 2020, now()),
        (NULL, '3', '1', '250', '1', 2020, now()),
        (NULL, '3', '1', '250', '2', 2020, now()),
        (NULL, '1', '2', '200', '1', 2020, now()),
        (NULL, '1', '2', '200', '2', 2020, now()),
        (NULL, '1', '2', '200', '3', 2020, now()),
        (NULL, '4', '2', '200', '1', 2020, now()),
        (NULL, '4', '2', '200', '2', 2020, now());
```

Задача 2. Напишете заявка, в която демонстрирате SELECT с ограничаващо условие по избор - ще изведем информация за всички ученици, които са с адрес в ж.к. Младост:

```
SELECT * FROM students
WHERE address LIKE '%Mladost%';
```

id	name	egn	address	phone	class
1	Iliyan Ivanov	9401150045	Sofia-Mladost 1	0893452120	10
3	Elena Petrova Petrova	9505052154	Sofia-Mladost 3	0897852412	11
4	Ivan Iliev Iliev	9510104542	Sofia-Mladost 3	0894123457	11
5	Maria Hristova Dimova	9510104547	Sofia-Mladost 4	0894123442	11
NULL	NULL	NULL	NULL	NULL	NULL

Задача 3. Напишете заявка, в която използвате агрегатна функция и GROUP BY по ваш избор – ще изведем средните стойности на платените такси за всяка една от групите:

```
SELECT group_id AS GroupId, AVG(paymentAmount) AS AvgOfAllPaymentPerGroup
FROM taxespayments
GROUP BY group_id;
```

GroupId	AvgOfAllPaymentPerGroup
1	232.69230769230768
2	200

Задача 4. Напишете заявка, в която демонстрирате INNER JOIN по ваш избор – ще напишем заявка, с която ще изведем цялата информация за всички групи и спорта, който тренират. Информацията се съхранява в таблиците sportGroups и sports:

```
SELECT sportgroups.location,
sportgroups.dayOfWeek,
sportgroups.hourOfTraining,
sportgroups.dayOfWeek,
sports.name
FROM sportgroups JOIN sports
ON sportgroups.sport_id = sports.id;
```

location	dayOfWeek	hourOfTraining	dayOfWeek	name
Sofia-Mladost 1	Monday	08:00:00	Monday	Football
Sofia-Mladost 1	Monday	09:30:00	Monday	Football
Plovdiv	Monday	12:00:00	Monday	Football
Sofia-Liylin 7	Sunday	08:00:00	Sunday	Volleyball
Sofia-Liylin 7	Sunday	09:30:00	Sunday	Volleyball

Задача 5. Напишете заявка, в която демонстрирате OUTER JOIN по ваш избор – ще напишем заявка, с която ще изведем цялата информация за всички групи и треньорите, които водят занятията. Информацията се съхранява в таблиците sportGroups и coaches. За свързването ще използваме RIGHT OUTER JOIN и по този начин – ако има треньори, за които все още няма разпределени групи, би трябвало да присъстват в резултатите:

```
SELECT sportgroups.location,
sportgroups.dayOfWeek,
sportgroups.hourOfTraining,
sportgroups.sport_id,
coaches.name
FROM sportgroups RIGHT OUTER JOIN coaches
ON sportgroups.coach_id = coaches.id;
```

location	dayOfWeek	hourOfTraining	sport_id	name
Sofia-Mladost 1	Monday	08:00:00	1	Ivan Todorov Petkov
Sofia-Liylin 7	Sunday	08:00:00	2	Ivan Todorov Petkov
Plovdiv	Monday	12:00:00	1	Ivan Todorov Petkov
Sofia-Mladost 1	Monday	09:30:00	1	georgi Ivanov Todorov
Sofia-Liylin 7	Sunday	09:30:00	2	georgi Ivanov Todorov
NULL	NULL	NULL	NULL	Ilian Todorov Georgiev
NULL	NULL	NULL	NULL	Petar Slavkov Yordanov
NULL	NULL	NULL	NULL	Todor Ivanov Ivanov
NULL	NULL	NULL	NULL	Slavi Petkov Petkov

Задача 6. Напишете заявка, в която демонстрирате вложен SELECT по ваш избор – ще създадем заявка, с която ще изведем всички треньори заедно със спортовете, на които водят тренировки. Информацията се намира в таблиците coaches и sports. Те не са свързани директно, а чрез трета таблица – sportGroups, което налага използването на вложен SELECT, за да извлечем информацията:

```
SELECT coaches.name,sports.name
from coaches JOIN sports
ON coaches.id IN (
                    SELECT coach_id
                    FROM sportgroups
                    WHERE sportgroups.sport_id = sports.id
                );
```

name	name
Ivan Todorov Petkov	Football
georgi Ivanov Todorov	Football
Ivan Todorov Petkov	Volleyball
georgi Ivanov Todorov	Volleyball

Задача 7. Напишете заявка, в която демонстрирате едновременно JOIN и агрегатна функция – ще изведем имената и сумите от платените такси на всички ученици по месеци през годините като използваме и ORDER BY, за да подредим резултатите по азбучен ред и ги ограничаваме до първите 15:

```
SELECT students.id, students.name AS StudentName, SUM(tp.paymentAmount)
AS SumOfAllPaymentPerGroup, tp.month AS Month
FROM taxespayments AS tp JOIN students
ON tp.student_id = students.id
GROUP BY month, student_id
ORDER BY StudentName
LIMIT 15;
```

id	StudentName	SumOfAllPaymentPerGroup	Month
3	Elena Petrova Petrova	500	7
3	Elena Petrova Petrova	500	10
3	Elena Petrova Petrova	500	8
3	Elena Petrova Petrova	500	6
3	Elena Petrova Petrova	500	11
3	Elena Petrova Petrova	500	9
3	Elena Petrova Petrova	500	5
3	Elena Petrova Petrova	500	12
3	Elena Petrova Petrova	500	4
3	Elena Petrova Petrova	500	3
3	Elena Petrova Petrova	750	2
3	Elena Petrova Petrova	750	1
1	Iliyan Ivanov	800	5
1	Iliyan Ivanov	800	6
1	Iliyan Ivanov	800	12

Задача 8. Създайте тригер по ваш избор – ще създадем един тригер, който прави лог на всички промени, направени по таблицата salarypayments:

```
delimiter |
CREATE TRIGGER after_salarypayment_update AFTER UPDATE ON salarypayments
FOR EACH ROW
BEGIN
INSERT INTO salarypayments_log(operation,
old_coach_id,
new_coach_id,
old_month,
new_month,
old_year,
new_year,
old_salaryAmount,
new_salaryAmount,
old_dateOfPayment,
new_dateOfPayment,
dateOfLog)
VALUES ('UPDATE',
OLD.coach_id,
CASE NEW.coach_id WHEN OLD.coach_id THEN NULL ELSE NEW.coach_id END,
OLD.month,
CASE NEW.month WHEN OLD.month THEN NULL ELSE NEW.month END,
OLD.year,
CASE NEW.year WHEN OLD.year THEN NULL ELSE NEW.year END,
OLD.salaryAmount,
CASE NEW.salaryAmount WHEN OLD.salaryAmount THEN NULL ELSE
NEW.salaryAmount END,
OLD.dateOfPayment,
CASE NEW.dateOfPayment WHEN OLD.dateOfPayment THEN NULL ELSE
NEW.dateOfPayment END,
NOW());
END;
|
Delimiter ;
```

Тестваме със следната ъпдейт заявка:

```
UPDATE `school_sport_clubs`.`salarypayments` SET `month`='4' WHERE
`id`='16';
```

Резултатът е запис в таблицата salarypayments_log:

id	operation	old_coach_id	new_coach_id	old_month	new_month	old_year	new_year	old_salaryAmount	new_salaryAmount	old_dateOfPayment	new_dateOfPayment	dateOfLog
1	UPDATE	1	NULL	2	4	2023	NULL	192	NULL	2023-03-05 17:01:01	NULL	2023-03-05 17:03:12
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Задача 9. Създайте процедура, в която демонстрирате използване на курсор – Ще създадем процедура, която проверява направените месечни часове за всеки треньор, записани в таблицата coach_work, изчислява дължимите хонорари по ставката, записана в таблицата coaches и записва сумата в таблицата salarypayments:

```
DELIMITER |
CREATE PROCEDURE monthHonorariumPayment(IN monthOfPayment INT, in
yearOFpayment INT)
procLabel: begin
DECLARE countOfCoaches int;
DECLARE iterator int;
DECLARE countOfRowsBeforeUpdate int;
DECLARE countOfRowsAfterUpdate int;
DECLARE finished int;
DECLARE tempCoachId int;
DECLARE tempSumOfHours int;

 DECLARE tempCoachCursor CURSOR FOR
    SELECT  coach_id, SUM(number_of_hours)
    FROM coach_work
    WHERE MONTH(coach_work.date) = monthOfPayment
    AND YEAR(coach_work.date) = yearOFpayment
    AND isPayed = 0
    GROUP BY coach_work.coach_id;

DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SELECT 'SQL Exception';
DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

CREATE TEMPORARY TABLE tempTbl(
    id int auto_increment primary key,
    coach_id int,
    number_of_hours int,
    pay_for_hour decimal,
    amount decimal,
    paymentMonth int,
    paymentYear int
)Engine = Memory;

       SELECT COUNT(*)
    INTO countOfRowsBeforeUpdate
    FROM coach_work
    WHERE MONTH(coach_work.date) = monthOfPayment
    AND YEAR(coach_work.date) = yearOFpayment
    AND isPayed = 0;

    START TRANSACTION;
        OPEN tempCoachCursor;
        SET finished = 0;
        while_loop_label: WHILE(finished = 0)
        DO
        FETCH tempCoachCursor INTO  tempCoachId, tempSumOfHours;

           IF(finished = 1)
```

```
                THEN leave while_loop_label;
                ELSE
                  SELECT  tempCoachId, tempSumOfHours;
                  INSERT INTO tempTbl(coach_id, number_of_hours,
pay_for_hour, amount, paymentMonth, paymentYear)
                  SELECT tempCoachId, tempSumOfHours, c.hour_salary,
tempSumOfHours*c.hour_salary, monthOfPayment, yearOFpayment
                      FROM coaches as c
                  WHERE c.id = tempCoachId;
              END IF;
          END WHILE;
          CLOSE tempCoachCursor;

    INSERT INTO salarypayments(`coach_id`,
`month`,`year`,`salaryAmount`,`dateOfPayment`)
          SELECT coach_id, paymentMonth, paymentYear, amount, NOW()
    FROM tempTbl
          ON DUPLICATE KEY UPDATE
          salaryAmount = salaryAmount + amount,
          dateOfPayment = NOW();

    UPDATE coach_work
    SET isPayed = 1
    WHERE month(coach_work.date) = monthOfPayment
    AND YEAR(coach_work.date) = yearOFpayment
    AND isPayed = 0;
    SELECT  ROW_COUNT() INTO countOfRowsAfterUpdate;
      IF(countOfRowsBeforeUpdate = countOfRowsAfterUpdate)
      THEN
        COMMIT;
      ELSE
        ROLLBACK;
      END IF;
    DROP TABLE tempTbl;
END;
|
DELIMITER ;
```

	id	coach_id	month	year	salaryAmount	dateOfPayment
▶	16	1	2	2023	192	2023-03-05 17:01:01
	17	2	2	2023	200	2023-03-05 17:01:01
*	NULL	NULL	NULL	NULL	NULL	NULL