ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ

Катедра „Компютърни системи”

КУРСОВ ПРОЕКТ

ПО БАЗИ ОТ ДАННИ

Студент: Васил Боянов Петров

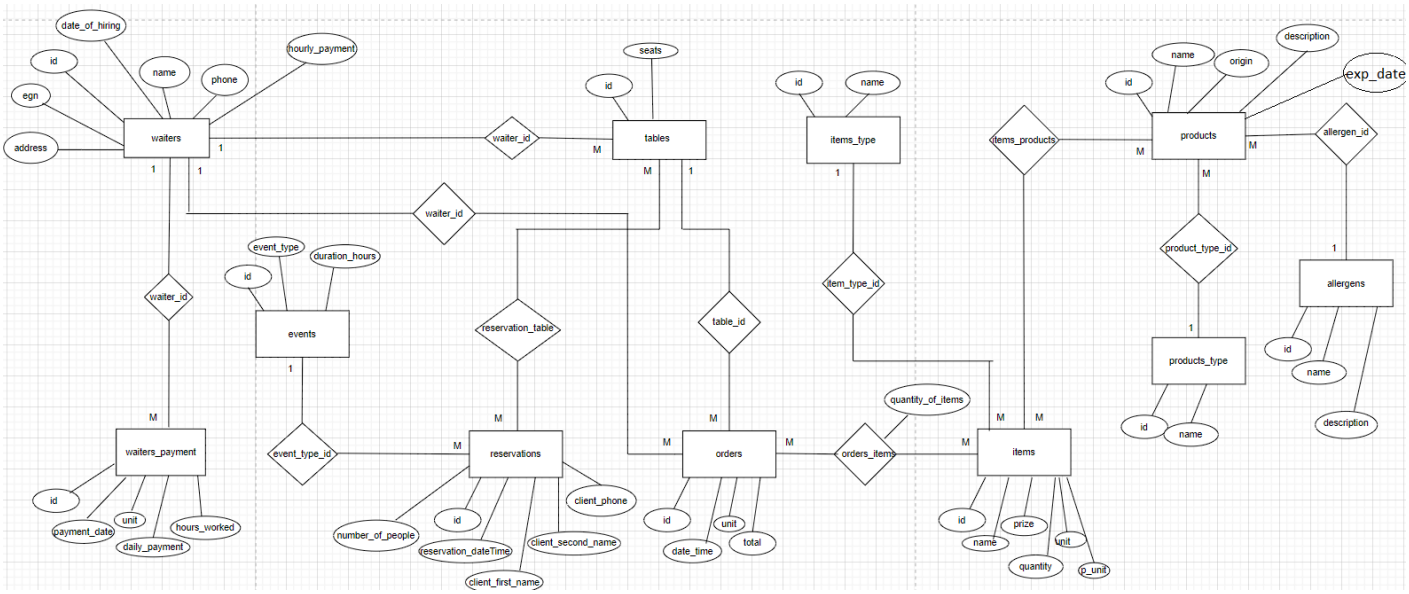
ФАК. № 121221084 Група: 43

Тема №21
Напишете база данни, пазеща информация за артикулите, които се предлагат от ресторант (напитки, храни). Да се пази информация за основните съставки, алергените, грамажа. Да се напише отделна таблица в която да се въвеждат поръчките, направени от клиентите (да се пази номер на масата, поръчка, сервитьор, дата и час и др.). 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.

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 ще съхраняваме информация са следните: Products, Items, Allergens, Orders, Waiters, Tables, Reservations, Events. Също така ще създадем и още няколко допълнителни таблици. Една от тях ще се казва Waiters_payment, в която ще пазим всичко свързано с плащанията на сервитьорите. Други две таблици са: Items_type и Products_type, тук ще съхраняваме типа на нашите, съответно артикули и продукти. Останалите допълнителни таблици са следните:
Reservation_table, Items_products и Orders_items. Те ни служат за представянето на връзките (M:M) между отделните основни таблици, като структурата им е идентична, с изключение на Orders_items, в която ще съхраняваме и количеството на съответния продукт в текущата поръчка (quantity_of_item). За проектирането на базата ще използваме модела ER-диаграма (Entity Relationship Diagram):



Заявките, с които създаваме базата данни и таблиците са:

```sql
CREATE DATABASE restaurant;

CREATE TABLE products_type(
    id INT AUTO_INCREMENT PRIMARY KEY,
   `name` VARCHAR(55) NOT NULL
);

CREATE TABLE allergens(
    id INT AUTO_INCREMENT PRIMARY KEY,
   `name` VARCHAR(55) NOT NULL,
   `description` TEXT NULL DEFAULT NULL
);

CREATE TABLE products(
    id INT AUTO_INCREMENT PRIMARY KEY,
   `name` VARCHAR(55) NOT NULL,
   `description` TEXT NULL DEFAULT NULL,
   origin VARCHAR(60) NULL DEFAULT NULL,
   UNIQUE(`name`, origin),
   product_type_id INT NOT NULL,
   CONSTRAINT FOREIGN KEY (product_type_id) REFERENCES products_type(id) ON DELETE
CASCADE ON UPDATE CASCADE,
   allergen_id INT NULL DEFAULT NULL,
   CONSTRAINT FOREIGN KEY (allergen_id) REFERENCES allergens(id) ON UPDATE CASCADE
);

CREATE TABLE items_type(
    id INT AUTO_INCREMENT PRIMARY KEY,
   `name` VARCHAR(55) NOT NULL
);

CREATE TABLE items(
    id INT AUTO_INCREMENT PRIMARY KEY,
   `name` VARCHAR(70) NOT NULL UNIQUE,
   quantity INT NOT NULL,
   unit ENUM('g', 'ml') NOT NULL,
   price DOUBLE NOT NULL CONSTRAINT PositivePrice CHECK (price > 0),
   p_unit VARCHAR(3) DEFAULT 'BGN' NOT NULL,
   item_type_id INT NOT NULL,
   CONSTRAINT FOREIGN KEY (item_type_id) REFERENCES items_type(id) ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE waiters(
    id INT AUTO_INCREMENT PRIMARY KEY,
   `name` VARCHAR(255) NOT NULL,
   egn CHAR(10) NOT NULL UNIQUE CONSTRAINT EGN CHECK(CHAR_LENGTH(egn) = 10),
   phone VARCHAR(20) NULL DEFAULT NULL,
   address VARCHAR(255) NULL DEFAULT NULL,
   date_of_hiring DATE NOT NULL,
   hourly_payment DOUBLE NOT NULL CONSTRAINT HPMoreThan4 CHECK(hourly_payment > 4),
   unit VARCHAR(3) DEFAULT 'BGN' NOT NULL
);
```

```
CREATE TABLE waiters_payment(
    id INT AUTO_INCREMENT PRIMARY KEY,
    payment_date DATE NOT NULL,
    hours_worked TINYINT NOT NULL CONSTRAINT PositiveHoursWorked CHECK(hours_worked > 0),
    daily_payment DOUBLE NOT NULL CONSTRAINT PositiveDailyPayment CHECK(daily_payment >0),
    unit VARCHAR(3) DEFAULT 'BGN' NOT NULL,
    waiter_id INT NOT NULL,
    CONSTRAINT FOREIGN KEY (waiter_id) REFERENCES waiters(id) ON DELETE CASCADE ON
UPDATE CASCADE,
    UNIQUE KEY (payment_date, waiter_id)
);

CREATE TABLE `tables`(
    id INT AUTO_INCREMENT PRIMARY KEY,
    seats TINYINT NOT NULL,
    waiter_id INT NULL DEFAULT NULL,
    CONSTRAINT FOREIGN KEY (waiter_id) REFERENCES waiters(id) ON DELETE SET NULL ON
UPDATE CASCADE
);

CREATE TABLE orders(
    id INT AUTO_INCREMENT PRIMARY KEY,
    date_time DATETIME NOT NULL,
    total DOUBLE NOT NULL,
    unit VARCHAR(3) DEFAULT 'BGN' NOT NULL,
    table_id INT NOT NULL,
    CONSTRAINT FOREIGN KEY (table_id) REFERENCES `tables`(id) ON DELETE CASCADE ON
UPDATE CASCADE,
    waiter_id INT NOT NULL,
    CONSTRAINT FOREIGN KEY (waiter_id) REFERENCES waiters(id) ON DELETE CASCADE ON
UPDATE CASCADE
);

CREATE TABLE items_products(
        item_id INT NOT NULL,
    CONSTRAINT FOREIGN KEY (item_id) REFERENCES items(id) ON DELETE CASCADE ON
UPDATE CASCADE,
    product_id INT NOT NULL,
    CONSTRAINT FOREIGN KEY (product_id) REFERENCES products(id) ON DELETE CASCADE ON
UPDATE CASCADE,
    UNIQUE(item_id, product_id)
);

CREATE TABLE orders_items(
    order_id INT NOT NULL,
    CONSTRAINT FOREIGN KEY (order_id) REFERENCES orders(id) ON DELETE CASCADE ON
UPDATE CASCADE,
    item_id INT NOT NULL,
    CONSTRAINT FOREIGN KEY (item_id) REFERENCES items(id) ON DELETE CASCADE ON
UPDATE CASCADE,
    quantity_of_item TINYINT NOT NULL DEFAULT 1,
    UNIQUE(order_id, item_id)
);

CREATE TABLE `events`(
        id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
    event_type ENUM("Рожден ден", "Бал", "Юбилей", "Вечеря", "Обяд", "Закуска", "Сватба",
"Друго събитие") NOT NULL UNIQUE,
    duration_hours INT NOT NULL
);
```

```
CREATE TABLE reservations(
   id INT AUTO_INCREMENT PRIMARY KEY,
   reservation_dateTime DATETIME NOT NULL CONSTRAINT CHECK (TIME(reservation_dateTime) >
'08:00:00' OR TIME(reservation_dateTime) < '23:00:00'),
   client_first_name VARCHAR(60) NOT NULL,
   client_second_name VARCHAR(60) NOT NULL,
   client_phone VARCHAR(15) NOT NULL,
   number_of_people TINYINT NOT NULL,
   event_type_id INT NOT NULL,
   CONSTRAINT FOREIGN KEY (event_type_id) REFERENCES `events`(id) ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE reservation_table(
    reservation_id INT NOT NULL,
    CONSTRAINT FOREIGN KEY (reservation_id) REFERENCES reservations(id) ON DELETE CASCADE
ON UPDATE CASCADE,
    table_id INT NOT NULL,
    CONSTRAINT FOREIGN KEY (table_id) REFERENCES `tables`(id) ON DELETE CASCADE ON
UPDATE CASCADE,
    UNIQUE KEY (reservation_id, table_id)
);
```

Добавяме и тестови данни в таблиците:

```
INSERT INTO products_type(`name`)
VALUES
('Зеленчуци'),
('Плодове'),
('Месо'),
('Морска храна'),
('Яйца'),
('Млечни продукти'),
('Подправки'),
('Ядки'),
('Зърнени култури');

INSERT INTO allergens (`name`, `description`) VALUES
('Глутен', 'Протеин, съдържащ се в пшеницата, ечемика и ръжта'),
('Соя', 'Бобово растение, често използвано като съставка в много преработени храни'),
('Фъстъци', 'Вид ядка, която може да предизвика тежки алергични реакции при някои
хора'),
('Дървесни ядки', 'Различни видове ядки, като бадеми, кашу и орехи'),
('Лактоза', 'Често срещана съставка в много млечни продукти, включително сирене,
кисело мляко и сладолед'),
('Яйца', 'Често срещана съставка в много печени изделия, включително торти и
бисквити'),
('Морски дарове', 'Различни видове морски дарове, като скариди, раци и омари'),
('Риба', 'Често срещана съставка в много видове кухня'),
('Сусам', 'Семе, което обикновено се използва като съставка в много видове хляб и
сладкиши'),
('Пшеница', 'Универсална съставка, която се използва в голямо разнообразие от храни,
включително хляб, макаронени изделия, зърнени храни и печени изделия'),
('Сулфити', 'Вид консервант, който често се използва в много видове преработени
храни');
```

```
INSERT INTO products (`name`, `description`, origin, product_type_id, allergen_id) VALUES
('Моркови', 'Кореноплоден зеленчук, който често се консумира суров или сготвен',
'Европа', 1, NULL),
('Домати', 'Плод, който често се използва като зеленчук в кулинарията', 'Европа', 1, NULL),
('Чушки', 'Група зеленчуци с разнообразни цветове и вкусове', 'Европа', 1, NULL),
('Лук', 'Луковичен зеленчук, който често се използва като ароматизатор в кулинарията',
'Европа', 1, NULL),
('Картофи', 'Кореноплоден зеленчук с нишесте, който често се вари, пече или пържи',
'Европа', 1, NULL),
('Краставици', 'Вид тиква, която често се консумира сурова или маринована', 'Азия', 1,
NULL), …

INSERT INTO items_type(`name`) VALUES
('Тестени изделия'),
('Предястия'),
('Основни ястия'),
('Десерти'),
('Алкохолни напитки'),
('Безалкохолни напитки');

INSERT INTO items(`name`, quantity, unit ,prize, item_type_id) VALUES
('Филия хляб', 70, ‘g’, 0.15, 1),
('Питка със сирене', 110, ‘g’, 1.50, 1),
('Пърленка', 90, ‘g’, 1.10, 1),
('Шопска салата', 300, ‘g’, 5.50, 2),
('Гръцка салата', 350, ‘g’, 7.00, 2),
('Цезар салата', 250, ‘g’, 6.00, 2),
('Млечна салата', 100, ‘g’, 2.50, 2),
('Таратор', 150, ‘g’, 2.00, 2),
('Пълнена чушка със сирене', 150, ‘g’, 3.00, 2),
('Пататник', 200, ‘g’, 4.50, 2),
('Пилешки хапки', 150, ‘g’, 7.00, 2),
…

INSERT INTO items_products VALUES
(1, 48),
(2, 48),
(2, 27),(3, 48),(4, 6),(4, 2),(4, 3),(4, 27),(4, 32),(4, 34),(5, 2),(5, 6),
(5, 3),(5, 4),(5, 34),(5, 49),(5, 32),(6, 7),(6, 1),(6, 27),(6, 30),(6, 49),
(7, 6),(7, 49),(7, 24),(7, 34),(8, 6),(8, 49),(8, 24),(9, 3),(9, 27),(10, 5),
(10, 4),(10, 35),(10, 30),(10, 48),(10, 34),(10, 32),(10, 49),(10, 37),(11, 15),(11, 48),
…
INSERT INTO waiters (`name`, egn, phone, address, date_of_hiring, hourly_payment)
VALUES
('Мария Петрова', '0243064568', '0888193275', 'бул. България 25', '2020-03-10', 8.20),
('Димитър Георгиев', '0343206467', NULL, 'ул. Христо Ботев 12', '2019-08-05', 13.00),
('София Димитрова', '9903148563', '0891849302', 'ул. Георги Бенковски 2', '2022-02-28',
7.00),
('Николай Лазаров', '9810294577', '0892927321', NULL, '2021-06-10', 7.80),
('Гергана Стоянова', '0145144567', NULL, 'бул. Цариградско шосе 55', '2020-01-01', 9.00),
('Станислава Маринова', '9805134547', '0894820472', 'ул. Любен Каравелов 8', '2021-12-
01', 8.50);
('Иван Иванов', '9503028563', '0888123456', 'ул. България 15, София', '2020-05-01', 10.50);
('Петър Петров', '9901267504', '0899654321', 'бул. Цар Борис III 25, Варна', '2019-01-15',
9.00);
('Мария Георгиева', '9812064567', NULL, 'ул. Гео Милев 18, Пловдив', '2021-09-01', 11.00);
```

```
INSERT INTO tables (seats, waiter_id) VALUES
(2, NULL),
(2, NULL),
(4, NULL),
(8, 5),
(4, NULL),
(4, NULL),
(6, NULL),
(10, 8),
(10, 2);

INSERT INTO orders(date_time, table_id, waiter_id) VALUES
('2021-09-01', 1, 2),
('2021-09-04', 2, 7),
('2021-09-06', 4, 6),
('2021-09-09', 7, 5),
('2021-09-11', 6, 9),
('2021-09-13', 3, 3);

INSERT INTO orders_items VALUES
(1, 1, 6),(1, 4, 1),(1, 5, 1),(1, 13, 1),(1, 20, 1),(1, 27, 2),\
(2, 3, 2),(2, 2, 2),(2, 6, 2),(2, 5, 1),(2, 4, 1),(2, 12, 1),(2, 22, 1), (2, 17, 1),(2, 16, 1),(2, 29, 4),
(3, 1, 10),(3, 7, 2),(3, 8, 2),(3, 14, 10),(3, 15, 10),(3, 23, 2),(3, 24, 2),(3, 36, 2),(3, 30, 2),
(4, 1, 25),(4, 9, 5),(4, 11, 5),(4, 21, 4),(4, 18, 3),(4, 22, 3),(4, 26, 5),(4, 27, 8),(4, 28, 2),(5, 3,
6),(5, 2, 6),(5, 4, 3),(5, 7, 3),(5, 12, 2),(5, 13, 2),(5, 19, 2),(5, 26, 1),(5, 34, 1)(6, 1, 16),(6, 8,
3),(6, 9, 3),(6, 10, 2),(6, 18, 4),(6, 20, 2),(6, 17, 2),(6, 23, 3),(6, 33, 3),(6, 30, 3),(6, 28, 2);

INSERT INTO waiters_payment(payment_date, hours_worked, waiter_id) VALUES
('2022-01-10', 8, 1),
('2022-01-12', 7, 1),
('2022-01-14', 6, 1),
('2022-01-16', 5, 1),
('2022-01-23', 8, 1),
('2022-02-14', 9, 1),
('2022-02-17', 8, 1),
('2022-02-19', 8, 1),
('2022-02-25', 7, 1),
('2022-01-10', 8, 2),
('2022-01-12', 7, 2),
('2022-01-14', 6, 2),
('2022-01-16', 5, 2),
('2022-01-23', 8, 2),
('2022-02-14', 9, 2),
('2022-02-17', 8, 2),
('2022-02-19', 8, 2),
('2022-02-25', 7, 2),
…

INSERT INTO reservations(reservation_dateTime, client_first_name, client_second_name,
client_phone, number_of_people, event_type_id)  VALUES
('2023-04-06 12:00:00', 'Иван', 'Иванов', '0888643255', 2, 2),
('2023-04-06 12:00:00', 'Петър', 'Петров', '0897320121', 4, 2),
('2023-04-06 18:00:00', 'Мария', 'Иванова', '0876923757', 8, 3),
('2023-04-06 18:00:00', 'Георги', 'Георгиев', '0896128462', 15, 3),
('2023-04-06 20:00:00', 'Анна', 'Петрова', '0876923786', 20, 4),
('2023-04-06 16:00:00', 'Васил', 'Петров', '0876823532', 25, 3);
```

```
INSERT INTO `events` VALUES
(NULL, "Закуска", 2),
(NULL, "Обяд", 3),
(NULL, "Вечеря", 3),
(NULL, "Рожден ден", 5),
(NULL, "Бал", 6),
(NULL, "Юбилей", 7),
(NULL, "Сватба", 8),
(NULL, "Друго събитие", 4);
```

Задача 2. Напишете заявка, в която демонстрирате SELECT с ограничаващо условие по избор – ще изведем информация за всички продукти, на които произхода им е от Европа:

```
SELECT * FROM products
WHERE origin = 'Европа';
```

id	name	description	origin	product_type_id	allergen_id
1	Моркови	Кореноплоден зеленчук, който често се кон...	Европа	1	NULL
2	Домати	Плод, който често се използва като зеленчу...	Европа	1	NULL
3	Чушки	Група зеленчуци с разнообразни цветове и в...	Европа	1	NULL
4	Лук	Луковичен зеленчук, който често се използ...	Европа	1	NULL
5	Картофи	Кореноплоден зеленчук с нишесте, който че...	Европа	1	NULL
10	Портокали	Цитрусов плод със сочна месеста част	Европа	2	NULL
14	Свинско месо	Вид месо, което се добива от прасета	Европа	3	NULL
23	Баракуда	Маслена риба със силна ароматна нотка и ме...	Европа	4	8
24	Кисело мляко	Млечен продукт, който се получава чрез за...	Европа	6	5
25	Прясно мляко	Непастеризирано, незагрявано мляко, получ...	Европа	6	5
26	Сметана	Млечен продукт, който се получава от отде...	Европа	6	5
27	Сирене	Млечен продукт, който се получава чрез об...	Европа	6	5
28	Масло	Млечен продукт, който се получава чрез от...	Европа	6	5
29	Кашкавал	Млечен продукт, който се получава чрез об...	Европа	6	5
37	Магданоз	Билка, която се използва като подправка в ...	Европа	7	NULL
38	Кимион	Подправка, която се получава от семената ...	Европа	7	NULL
45	Пшеница	Една от най-широко разпространените зърн...	Европа	9	1
48	Брашно	прахообразен продукт, който се получава о...	Европа	9	1
50	Кисело зеле	Хранителен продукт, който се получава чре...	Европа	1	NULL
NULL	NULL	NULL	NULL	NULL	NULL

Задача 3. Напишете заявка, в която използвате агрегатна функция и GROUP BY по ваш избор - ще изведем произходите на продуктите и тяхната съответна бройка:

```
SELECT origin AS ProductOrigin,
COUNT(*) AS NumOfProductsFromThatOrigin
FROM products
GROUP BY origin;
```

	ProductOrigin	NumOfProductsFromThatOrigin
▶	Азия	13
	Северна Америка	7
	Южна Америка	5
	Европа	19
	Африка	6

Задача 4. Напишете заявка, в която демонстрирате INNER JOIN по ваш избор – заявката ще извежда информация за всеки един артикул и неговия тип, за целта ще обединим информация от таблиците items и items_type:

```
SELECT items.name AS Item,
items.quantity AS QuantityOfItem,
items.unit AS UnitOfItem,
items.prize AS PrizeOfItem,
items.p_unit AS PrizeUnitOfItem,
items_type.name AS TypeOfItem
FROM items JOIN items_type
ON items.item_type_id = items_type.id;
```

	Item	QuantityOfItem	UnitOfItem	PrizeOfItem	PrizeUnitOfItem	TypeOfItem
▶	Агнешко с яйца	350	g	17	BGN	Основни ястия
	Газирана вода	500	ml	1.5	BGN	Безалкохолни н...
	Гръцка салата	350	g	7	BGN	Предястия
	Гювеч	450	g	10	BGN	Основни ястия
	Кафе	50	ml	2.5	BGN	Безалкохолни н...
	Кебапче	80	g	1.2	BGN	Основни ястия
	Кисело зеле със свинско м...	350	g	11.5	BGN	Основни ястия
	Кока-кола	330	ml	2.5	BGN	Безалкохолни н...
	Крем карамел	80	g	5.5	BGN	Десерти
	Кюфте	70	g	1.1	BGN	Основни ястия
	Млечна салата	100	g	2.5	BGN	Предястия
	Мусака	300	g	7.5	BGN	Основни ястия
	Наливна бира	500	ml	2	BGN	Алкохолни напи...
	Пататник	200	g	4.5	BGN	Предястия
	Патешка магданозова чорба	250	g	13.5	BGN	Основни ястия
	Печено агнешко с картофи	450	g	15.5	BGN	Основни ястия
	Печено свинско с картофи	450	g	13.5	BGN	Основни ястия

Задача 5. Напишете заявка, в която демонстрирате OUTER JOIN по ваш избор - заявката ще извежда информацията от таблицата tables, като също така ще се изведе и името на сервитьора, който има персонална задача да обслужва тази маса, за целта ще използваме LEFT OUTER JOIN, като ще се покаже информацията и за масите, които нямат строго определен сервитьор за тях:

```
SELECT `tables`.id AS TableNumber,
waiters.name AS WaiterName
FROM `tables` LEFT JOIN waiters
ON `tables`.waiter_id = waiters.id;
```

TableNumber	WaiterName
1	NULL
2	NULL
3	NULL
4	Гергана Стоянова
5	NULL
6	NULL
7	NULL
8	Петър Петров
9	Димитър Георгиев

Задача 6. Напишете заявка, в която демонстрирате вложен SELECT по ваш избор - със следващата заявка ще изведем информация за поръчките и съответните артикули, които са включени в тях. За целта ще използваме таблиците orders и items, както и свързващата ги таблица orders_items:

```
SELECT orders_items.order_id AS OrderNumber,
items.`name` AS ItemInTheOrder,
orders_items.quantity_of_item AS QuantityOfItem
FROM orders_items JOIN items
ON orders_items.order_id IN (
    SELECT id
    FROM orders
)
AND orders_items.item_id = items.id;
```

	OrderNumber	ItemInTheOrder	QuantityOfItem
▶	1	Филия хляб	6
	1	Шопска салата	1
	1	Гръцка салата	1
	1	Мусака	1
	1	Агнешко с яйца	1
	1	Наливна бира	2
	2	Питка със сирене	2
	2	Пърленка	2
	2	Шопска салата	1
	2	Гръцка салата	1
	2	Цезар салата	2
	2	Гювеч	1
	2	Печено агнешко ...	1
	2	Риба на скара	1
	2	Спагети	1
	2	Ракия	4
	3	Филия хляб	10
	3	Млечна салата	2

Задача 7. Напишете заявка, в която демонстрирате едновременно JOIN и агрегатна функция - ще изведем информация за всеки един сервитьор и неговото месечно възнаграждение, подредено по намаляващ ред и ще ограничим резултата до първите 9 записа:

```
SELECT waiter_id AS WaiterId, `name` AS WaiterName, MONTH(payment_date) AS Month,
SUM(daily_payment) AS MontlyEarnings
FROM waiters_payment JOIN waiters
ON waiter_id = waiters.id
GROUP BY waiter_id, MONTH(payment_date)
ORDER BY MontlyEarnings DESC
LIMIT 9;
```

	WaiterId	WaiterName	Month	MontlyEarnings
▶	2	Димитър Георгиев	1	442
	2	Димитър Георгиев	2	416
	9	Мария Георгиева	1	374
	7	Иван Иванов	1	357
	9	Мария Георгиева	2	352
	7	Иван Иванов	2	336
	8	Петър Петров	1	306
	5	Гергана Стоянова	1	306
	6	Станислава Маринова	1	289

Задача 8. Създайте тригер по ваш избор – ще създадем тригер, който преди въвеждане на данни в таблицата reservations, ще проверява дали е възможно тази резервация да бъде направена, според вече наличните резервации в таблицата:

```
delimiter |
DROP TRIGGER IF EXISTS constraint_for_reservation;
CREATE TRIGGER constraint_for_reservation BEFORE INSERT ON reservations
FOR EACH ROW
BEGIN
   DECLARE bef_event_seats TINYINT;
   DECLARE res_id INT;
   DECLARE diff_hours TINYINT;
   DECLARE bef_event_type_id INT;
   DECLARE bef_event_hours TINYINT;
   DECLARE count_of_rows INT;
   DECLARE iterator INT;
   DECLARE date_time_res_closing DATETIME;
   DECLARE event_seats TINYINT;
   DECLARE flag TINYINT;
   DECLARE new_event_hours TINYINT;
   DECLARE not_avail_seats INT;

   SELECT duration_hours INTO new_event_hours
   FROM `events`
   WHERE id = NEW.event_type_id;

   SET @result = 0;
   SET @currSeats = NEW.number_of_people;
   SET @bef_seats_sum = 0;
   SET flag = 1;
   SET not_avail_seats = 0;

   SET date_time_res_closing= STR_TO_DATE(CONCAT(DATE(DATE_ADD(
NEW.reservation_dateTime, INTERVAL 1 DAY)), ' ', '02:00:00'), '%Y-%m-%d %H:%i:%s');

   IF (DATE_ADD(NEW.reservation_dateTime, INTERVAL new_event_hours HOUR) >
date_time_res_closing)
   THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "Съжалявам, но ресторанта ще
бъде затворен докато вашето събитие приключи!";
   END IF;
   CALL available_tables(NEW.reservation_dateTime, NEW.id, @currSeats, @result);

   CREATE TEMPORARY TABLE tempRes(
      id INT,
      difference_hours INT,
      bef_e_type_id INT
   )ENGINE = MEMORY;

   CREATE TEMPORARY TABLE tempTabBefEvents(
      tempTabBefEvenetId INT,
      tempTabSeats TINYINT
   )ENGINE = MEMORY;

   IF (@result = 0)
   THEN
```

```
        INSERT INTO tempRes
                SELECT id, TIMESTAMPDIFF(HOUR, reservation_dateTime,
NEW.reservation_dateTime), event_type_id
                FROM reservations
                WHERE DATE(reservation_dateTime) = DATE(NEW.reservation_dateTime)
                ORDER BY CASE WHEN reservation_dateTime <= NEW.reservation_dateTime
THEN 0 ELSE 1 END ASC,
                ABS(TIMESTAMPDIFF(HOUR, reservation_dateTime,
NEW.reservation_dateTime)) ASC,  reservation_dateTime ASC;

        SELECT COUNT(id) INTO count_of_rows
        FROM tempRes;

        SET iterator = 1;
      res_loop: WHILE (iterator >= 1 AND iterator <= count_of_rows)
        DO
         SELECT id, difference_hours, bef_e_type_id INTO
                                                    res_id , diff_hours, bef_event_type_id
         FROM tempRes
         LIMIT 1;

         DELETE FROM tempRes
         WHERE id = res_id;

         SELECT duration_hours INTO bef_event_hours
         FROM `events`
         WHERE id = bef_event_type_id;

         IF (diff_hours > 0 AND diff_hours >= bef_event_hours)
         THEN
             IF (flag = 1)
            THEN
                CALL events_sum(@bef_seats_sum, res_id);
                IF (@currSeats + @bef_seats_sum >= NEW.number_of_people)
                THEN
                        SET flag = 0;
                END IF;
            END IF;
         ELSE IF (diff_hours <= 0 AND ABS(diff_hours) >= new_event_hours)
         THEN
             IF (flag = 1)
             THEN
                CALL events_sum(@bef_seats_sum, res_id);
                IF (@currSeats + @bef_seats_sum >= NEW.number_of_people)
                THEN
                        LEAVE res_loop;
                END IF;
             END IF;
         ELSE
                SELECT number_of_people INTO event_seats
                FROM reservations
                WHERE id = res_id;

                SET not_avail_seats = not_avail_seats + event_seats;

         IF (not_avail_seats + NEW.number_of_people > (SELECT SUM(seats) FROM `tables`))
```

```
                THEN
                        DROP TEMPORARY TABLE IF EXISTS tempRes;
                        DROP TEMPORARY TABLE IF EXISTS tempTables;
                        DROP TEMPORARY TABLE IF EXISTS tempTabBefEvents;
                        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "Нямаме толкова
свободни места за вашата резервация за този час, те вече са запазени за друга такава!";
                END IF;
                INSERT INTO tempTabBefEvents (tempTabBefEvenetId)
                        SELECT table_id
                        FROM reservation_table
                        WHERE reservation_id = res_id;
        END IF;
        END IF;
    SET iterator = iterator + 1;
    END WHILE;
    IF ((iterator - 1 = count_of_rows AND flag = 1))
    THEN
        DROP TEMPORARY TABLE IF EXISTS tempRes;
        DROP TEMPORARY TABLE IF EXISTS tempTables;
        DROP TEMPORARY TABLE IF EXISTS tempTabBefEvents;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = "Нямаме толкова много
свободни места за вашата резервация за този час!";
    END IF;
  END IF;
  DROP TEMPORARY TABLE IF EXISTS tempRes;
END;
|
delimiter ;
```

Ще създадем и тригер, който след успешно въведена резервация в таблицата reservations, ще въвежда в таблицата reservation_table, съответната резервация заедно с нейните резервирани маси:

```
delimiter |
DROP TRIGGER IF EXISTS insert_into_reservations_tables;
CREATE TRIGGER insert_into_reservations_tables AFTER INSERT ON reservations
FOR EACH ROW
BEGIN
                CALL update_reser_tab(NEW.id, NEW.number_of_people);
END;
|
delimiter ;
```

Задача 9. Създайте процедура, в която демонстрирате използване на курсор – ще създадем няколко процедури, които ще ни помогнат в процеса на резервации, като първата от тях ще бъде процедура, която проверява дали има достатъчно свободни маси за съответната резервация:

```
delimiter |
DROP PROCEDURE IF EXISTS available_tables;
CREATE PROCEDURE available_tables(IN res_dateTime DATETIME, IN res_id INT,
                                  INOUT res_seats TINYINT, OUT res TINYINT)
BEGIN
        DECLARE finished INT;
        DECLARE currTableId INT;
        DECLARE currAvaSeats1 TINYINT;
        DECLARE tableCursor CURSOR FOR
        SELECT id
        FROM `tables`
        ORDER BY seats;

        DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

        SET currAvaSeats1 = 0;
        SET finished = 0;

        CREATE TEMPORARY TABLE tempTables(
                table_id INT,
                table_seats TINYINT
        )ENGINE = MEMORY;

        OPEN tableCursor;
        table_loop: WHILE(finished = 0)
        DO
           FETCH tableCursor INTO currTableId;
           IF(finished = 1)
          THEN
                LEAVE table_loop;
          END IF;

          INSERT INTO tempTables
                SELECT id, seats
                FROM `tables`
                WHERE id = currTableId
                AND id NOT IN (
                        SELECT table_id
                        FROM reservation_table
                        WHERE reservation_id IN (
                                SELECT id
                                FROM reservations
                                WHERE DATE(reservation_dateTime) = DATE(res_dateTime)
                        )
                );

        SELECT SUM(table_seats) INTO currAvaSeats1
        FROM tempTables;

END WHILE;
CLOSE tableCursor;
```

```
SET currAvaSeats1 = IFNULL(currAvaSeats1, 0);
IF (currAvaSeats1 < res_seats)
THEN
        DELETE FROM tempTables;
        SET res = 0;
         SET res_seats = currAvaSeats1;
ELSE
        SET res = 1;
END IF;
END;
|
delimiter ;
```

Втората процедура, ще ни връща сумата на местата на свободните маси за съответната резервация:

```
delimiter |
DROP PROCEDURE IF EXISTS events_sum;
CREATE PROCEDURE events_sum(OUT sum_of_seats INT, IN res_id INT)
BEGIN
        INSERT INTO tempTables
                SELECT id, seats
                FROM `tables`
                WHERE id IN (
                        SELECT table_id
                        FROM reservation_table
                        WHERE reservation_id = res_id
                )
                AND id NOT IN (
                        SELECT tempTabBefEvenetId
                        FROM tempTabBefEvents
                );

        SELECT SUM(table_seats) INTO sum_of_seats
        FROM tempTables;
END;
|
delimiter ;
```

Третата процедура, ще въвежда масите за съответната резервация в таблицата reservation_table:

```
delimiter |
DROP PROCEDURE IF EXISTS update_reser_tab;
CREATE PROCEDURE update_reser_tab(IN new_res_id INT, IN res_seats TINYINT)
BEGIN
   DECLARE tempTabId INT;
   DECLARE tempTabSeats TINYINT;
   DECLARE currSeats TINYINT;
   DECLARE table_id_min_seats INT;
   DECLARE table_min_seats TINYINT;
   DECLARE tabCursor CURSOR FOR
        SELECT table_id, table_seats FROM tempTables
        ORDER BY table_seats DESC;
```

```
    SET currSeats = 0;

    OPEN tabCursor;
    tab_loop: WHILE (TRUE)
    DO
        FETCH tabCursor INTO tempTabId, tempTabSeats;
        IF (currSeats + tempTabSeats >= res_seats)
        THEN
            SET table_id_min_seats = tempTabId;

            IF (currSeats + tempTabSeats - res_seats != 1 AND
                                                currSeats + tempTabSeats != res_seats)
            THEN
                SELECT table_id, table_seats INTO table_id_min_seats, table_min_seats
                FROM tempTables
                WHERE table_seats + currSeats >= res_seats
                ORDER BY table_seats ASC
                LIMIT 1;
                IF (res_seats < 5 AND currSeats + table_min_seats > 2 * res_seats)
                THEN
                    DROP TEMPORARY TABLE IF EXISTS tempRes;
                    DROP TEMPORARY TABLE IF EXISTS tempTables;
                    DROP TEMPORARY TABLE IF EXISTS tempTabBefEvents;
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT= "Съжалявам, но
свободните маси, които имаме в момента, са повече от два пъти по-големи от това, което
се опитвате да резервирате!";
                    END IF;
            END IF;

            INSERT INTO reservation_table
            VALUES (new_res_id, table_id_min_seats);
            LEAVE tab_loop;
        ELSE
            INSERT INTO reservation_table
            VALUES (new_res_id, tempTabId);

        END IF;
        SET currSeats = currSeats + tempTabSeats;
    END WHILE;
    CLOSE tabCursor;
    DROP TEMPORARY TABLE IF EXISTS tempTables;
    DROP TEMPORARY TABLE IF EXISTS tempTabBefEvents;
END;
|
delimiter ;
```

И последната четвърта процедура, ще изтрива резервации от таблицата reservations:

```
delimiter |
DROP PROCEDURE IF EXISTS delete_reservation;
CREATE PROCEDURE delete_reservation(IN res_id INT)
BEGIN
	DELETE FROM reservations
	WHERE id = res_id;
END;
|
delimiter ;
```

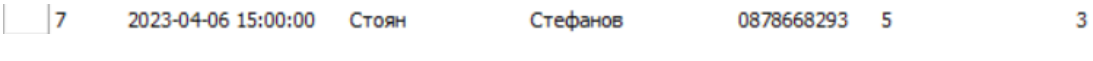
Ще тестваме със следните заявки:

```
INSERT INTO reservations(reservation_dateTime, client_first_name, client_second_name,
client_phone, number_of_people, event_type_id)
VALUES ('2023-04-06 15:00:00', 'Стоян', 'Стефанов', '0878668293', 5, 3);

INSERT INTO reservations(reservation_dateTime, client_first_name, client_second_name,
client_phone, number_of_people, event_type_id)
VALUES ('2023-04-06 13:00:00', 'Стоян', 'Стефанов', '0878668293', 70, 3);

INSERT INTO reservations(reservation_dateTime, client_first_name, client_second_name,
client_phone, number_of_people, event_type_id)
VALUES ('2023-04-06 22:00:00', 'Стоян', 'Стефанов', '0878668293', 10, 4);
```

Резултатът от следните заявки:



reservation_id	table_id
7	1
7	3