

Базы данных MySQL и PostgreSQL

Сафаргулов Ильнар

Представления

Представления MySQL “Книжное дело”

1. Создайте представление, которое показывает код поставки, наименование книги, дату поставки, наименование поставщика, стоимость поставки, объем поставки.

```
CREATE VIEW BooksDelivery AS
SELECT p.Code_purchase, b.Title_book, p.Date_order, d.Name_company, p.Cost, p.Amount FROM purchases p
JOIN books b ON p.Code_book = b.Code_book
JOIN deliveries d ON p.Code_delivery = d.Code_delivery;
```

Запрос создает представление, которое объединяет данные о заказах книг из таблиц: purchases берем номер заказа, дату, стоимость и количество; books берем название книги по совпадающему коду; deliveries добавляем название компании-доставщика. Получается готовая сводка с полной информацией о каждом заказе: что, когда, сколько, по какой цене и кто доставлял.

	Code_purchase	Title_book	Date_order	Supplier_Name	Cost	Amount
▶	3	Pride and Prejudice	2024-03-26	ОАО Filin	113.172	60
	6	War and Peace	2024-08-19	Company 8	66.36	10
	7	Crime and Punishment	2023-12-01	ОАО KushTau	32.652	19
	14	War and Peace	2017-06-28	Company 2	94.848	110
	24	Crime and Punishment	2024-06-19	Company 8	117.732	80
	27	The Odyssey	2014-02-11	Company 2	53.724	10
	38	To Kill a Mockingbird	2024-06-12	Company 8	106.584	30
	40	1984	2015-10-16	Company 3	13.032	22
	46	1984	2024-07-03	Company 6	14.688	15
	300	Moby Dick	2023-12-26	ОАО Filin	1200	6

2. Создайте представление, которое показывает все сведения об издательствах из города Москва.

```
CREATE VIEW PublishersInMoscow AS
SELECT * FROM publishing_house
WHERE City = 'Moscow';
```

Запрос создает представление PublishersInMoscow, которое выбирает все данные из таблицы publishing_house только для тех издательств, которые расположены в Москве. Получается фильтрованная выборка, содержащая информацию о московских издательствах.

	Code_publish	Publish	City
▶	0	Nauka	Moscow
	302	HarperCollins	Moscow
	308	Macmillan	Moscow

3. Создайте представление, которое показывает код книги, наименование книги, автора, количество книг на складе, стоимость книг (максимальная стоимость).

```
CREATE VIEW InfoBook AS
SELECT b.Code_book, b.Title_book, b.Code_author, p.Cost, p.Amount FROM Books b
JOIN purchases p ON b.Code_book = p.Code_book;
```

Запрос создает представление InfoBook, объединяющее данные о книгах и их заказах В выборку попадают: код книги, название, автор, стоимость и количество, связанные по общему полю Code_book.

	Code_book	Title_book	Code_author	Cost	Amount
▶	4	Pride and Prejudice	104	113.172	60
	6	War and Peace	106	66.36	10
	9	Crime and Punishment	109	32.652	19
	6	War and Peace	106	94.848	110
	9	Crime and Punishment	109	117.732	80
	8	The Odyssey	108	53.724	10
	2	To Kill a Mockingbird	102	106.584	30
	3	1984	103	13.032	22
	3	1984	103	14.688	15
	5	Moby Dick	105	1200	6

4. Создайте представление, которое показывает топ 5 книг с максимальным количеством на складе (используйте предыдущее представление).

```
CREATE VIEW Top5Book AS
SELECT Code_book, Title_book, Code_author, Amount, Cost FROM InfoBook
ORDER BY Amount DESC LIMIT 5;
```

Запрос создает представление, которое выбирает из InfoBook 5 самых продаваемых книг, сортируя их по количеству в убывающем порядке. В результат выводятся: код книги, название, автор, количество и стоимость.

Result Grid Filter Rows: <input type="text"/> Export: Wrap Cell Content					
	Code_book	Title_book	Code_author	Amount	Cost
▶	6	War and Peace	106	110	94.848
	9	Crime and Punishment	109	80	117.732
	4	Pride and Prejudice	104	60	113.172
	2	To Kill a Mockingbird	102	30	106.584
	3	1984	103	22	13.032

Представления PostgreSQL “Успеваемость”

1.Создайте представление, которое показывает преподавателей, не принимающих экзамены.

lectors_not_examining X

General Definition Code Security SQL

```

1 SELECT code_lector,
2       name_lector,
3       post,
4       science
5 FROM lectors l
6 WHERE NOT (EXISTS ( SELECT 1
7                     FROM progress p
8                     WHERE p.code_lector = l.code_lector));

```

Запрос выбирает из таблицы lectors , у которых нет ни одного экзамена в таблице progress. Использую подзапрос для проверки отсутствия связей между таблицами по полю code_lector.

	code_lector integer	name_lector character varying (100)	post character varying (50)	science character varying (100)
1	5	Сафин А.М.	Преподаватель	БЖД
2	4	Павлов А.Р	Преподаватель	Программирование
3	3	Петров Савелий Яковлевич	Преподаватель	Русский язык

2.Создайте представление, которое показывает список групп специальности Программирование в компьютерных системах.

programming_groups

General Definition Code Security SQL

```

1  SELECT code_group,
2      name_group,
3      num_course,
4      name_speciality
5  FROM groups g
6  WHERE name_speciality = 'Программист';

```

Запрос выбирает группы из таблицы groups , где специальность - "Программист". Фильтрации по полю name_speciality

	code_group integer	name_group character varying (100)	num_course integer	name_speciality character varying (100)
1	1	22п-1	3	Программист
2	2	22п-2	3	Программист

3.Создайте представление, которое показывает код студента, ФИО студента, наименование группы, средний балл за весь период обучения.

student_average_grades

General Definition Code Security SQL

```


1  SELECT s.code_stud,
2      s.surname,
3      s.name,
4      s.lastname,
5      g.name_group,
6      avg(p.estimate) AS avg_grade
7  FROM students s
8      JOIN groups g ON s.code_group = g.code_group
9      LEFT JOIN progress p ON s.code_stud = p.code_stud
10 GROUP BY s.code_stud, s.surname, s.name, s.lastname, g.name_group;

```

Запрос вычисляет средний балл студентов по всем предметам. Оно объединяет данные из таблиц students groups и progress. Функция вычисляет среднее арифметическое всех оценок студента из таблицы progress

	code_stud integer	surname character varying (50)	name character varying (50)	lastname character varying (50)	name_group character varying (100)	avg_grade numeric
1	5	Гилязетдинов	Роберт	Рушанович	22п-1	4.0000000000000000
2	3	Бахонов	Эмир	Алиханович	23веб-1	4.0000000000000000
3	6	Чернов	Никита	Димович	22п-2	4.6666666666666667
4	1	Сафаргулов	Ильнар	Айратович	22п-1	4.0000000000000000
5	11	Иванов	Павел	Сергеевич	22п-2	[null]
6	4	Давлатов	Минир	Макпалович	23веб-1	3.5000000000000000
7	2	Феденев	Владислав	Юрьевич	22п-2	4.0000000000000000

4.Создайте представление, которое показывает список из 5 студентов с наибольшим средним баллом за весь период обучения.

 top_5_students

General Definition **Code** Security SQL

```

1  SELECT s.code_stud,
2      s.surname,
3      s.name,
4      s.lastname,
5      avg(p.estimate) AS avg_grade
6  FROM students s
7      JOIN progress p ON s.code_stud = p.code_stud
8  GROUP BY s.code_stud, s.surname, s.name, s.lastname
9  ORDER BY (avg(p.estimate)) DESC
10 LIMIT 5;
```

Запрос выводит топ-5 студентов с наивысшим средним баллом. Соединяет таблицы students и progress. Для учащегося рассчитывается средний балл по всем предметам. Результаты сортируются по убыванию балла. Оставляет только 5 верхних записей, формируя итоговый рейтинг успеваемости.

	code_stud integer	surname character varying (50)	name character varying (50)	lastname character varying (50)	avg_grade numeric
1	6	Чернов	Никита	Димович	4.6666666666666667
2	3	Бахонов	Эмир	Алиханович	4.0000000000000000
3	5	Гилязетдинов	Роберт	Рушанович	4.0000000000000000
4	2	Феденев	Владислав	Юрьевич	4.0000000000000000
5	1	Сафаргулов	Ильнар	Айратович	4.0000000000000000

Представление “ГОСТИНИЦА”

1. Представление "Постояльцы, проживающие в гостинице в данное время".

current_guests

General Definition Code Security SQL

```
1  SELECT passportnumber,  
2      fullname,  
3      roomid  
4  FROM guests g  
5  WHERE checkindate <= CURRENT_DATE AND checkoutdate >= CURRENT_DATE;
```

Запрос выбирает актуальных гостей, находящихся в гостинице на текущую дату. Проходит проверка что дата заезда меньше или равна текущей дате, а дата выезда - больше или равна ей. Возвращает номер паспорта, ФИО и номер комнаты каждого постояльца.

2. Представление "Свободные места": класс – номер – общее количество мест в номере – количество свободных мест.

free_rooms

General Definition Code Security SQL

```
1  SELECT r.class,  
2      r.roomid,  
3      r.capacity,  
4      r.capacity - count(g.roomid) AS freespaces  
5  FROM rooms r  
6  LEFT JOIN guests g ON r.roomid = g.roomid AND g.checkoutdate >= CURRENT_DATE  
7  GROUP BY r.roomid, r.capacity, r.class  
8  ORDER BY r.roomid;
```

Запрос рассчитывает количество свободных мест для каждого номера гостиницы. Определяет разницу между вместимостью номера и количеством проживающих учитывая актуальных дат выезда. Группирует данные по id, его классу и вместимости.

3. Представление "Счёт на оплату номера": сумма оплаты за номер (стоимость, умноженная на количество дней проживания) и общей стоимости оказанных услуг.

room_bill

General Definition Code Security SQL

```
1  SELECT g.passportnumber,  
2      g.roomid,  
3      (g.checkoutdate - g.checkindate)::numeric * rc.price AS roomcost,  
4      COALESCE(sum(ps.price * ps.quantity::numeric), 0::numeric) AS servicecost,  
5      (g.checkoutdate - g.checkindate)::numeric * rc.price + COALESCE(sum(ps.price * ps.quantity::numeric), 0::numeric) AS totalcost  
6  FROM guests g  
7  JOIN rooms r ON g.roomid = r.roomid  
8  JOIN roomclasses rc ON r.class::text = rc.class::text  
9  LEFT JOIN providedservices ps ON g.passportnumber::text = ps.guest::text AND g.checkindate = ps.checkindate  
10 GROUP BY g.passportnumber, g.roomid, g.checkindate, g.checkoutdate, rc.price;
```

Запрос вычисляет полную стоимость номера для гостя. Стоимость номера определяется умножением длительности проживания на цену категории. Услуги суммируются с учетом их количества. COALESCE обрабатывает случаи отсутствия услуг.

Хранимые процедуры

Хранимых процедур «Книжное дело».

1. Вывести все сведения о поставке (все поля таблицы Purchases), а также название книги (поле Title_book) с максимальной общей стоимостью (использовать поля Cost и Amount).

```
BEGIN
    SELECT p.*, b.Title_book
    FROM purchases p
    JOIN books b ON p.Code_book = b.Code_book
    WHERE (p.Cost * p.Amount) = (
        SELECT MAX(Cost * Amount)
        FROM purchases
    );
END //
DELIMITER ;
CALL GetMaxCostPurchase();
```

Процедура находит покупку с максимальной общей стоимостью. Таблицы purchases и books объединяются и выводятся.

Code_purchase	Code_book	Date_order	Code_delivery	Type_purchase	Cost	Amount	Title_book
14	6	2017-06-28	2	opt	94.848	110	War and Peace

2. Сосчитать количество книг определенного автора (ФИО автора является входным параметром).


```

DELIMITER //
CREATE PROCEDURE CountBooksByAuthor(IN authorName VARCHAR(30))
BEGIN
    SELECT COUNT(*) AS BookCount
    FROM books b
    JOIN authors a ON b.Code_author = a.Code_author
    WHERE a.name_author LIKE CONCAT('%', authorName, '%');
END //
DELIMITER ;
CALL CountBooksByAuthor('Tolstoy');

```

Процедура подсчитывает количество книг автора. Принимает имя автора и ищет совпадения в таблице authors, после чего считает связанные записи в таблице books.

3. Определить адрес определенного поставщика (Наименование поставщика является входным параметром, адрес поставщика – выходным параметром).

```

DELIMITER //
CREATE PROCEDURE GetSupplierAddress(
    IN supplierName VARCHAR(30),
    OUT supplierAddress VARCHAR(100)
)
BEGIN
    SELECT Address INTO supplierAddress
    FROM deliveries
    WHERE Name_company LIKE CONCAT('%', supplierName, '%')
    LIMIT 1;
END //
DELIMITER ;
CALL GetSupplierAddress('Filin', @address);
SELECT @address;

```

Процедура возвращает адрес поставщика по его. Если название компании содержит указанную строку, процедура возвращает первый найденный адрес.

@address
1 Main Street, City 1

4. Выполните операцию вставки в таблицу Books. Код книги должен увеличиваться автоматически на единицу.


```

DELIMITER //
> CREATE PROCEDURE AddNewBook(
    IN pTitle VARCHAR(45),
    IN pAuthorCode INT,
    IN pPages INT,
    IN pPublishCode INT
)
> BEGIN
    DECLARE nextCode INT;
    SELECT IFNULL(MAX(Code_book), 0) + 1 INTO nextCode FROM books;

    INSERT INTO books (Code_book, Title_book, Code_author, Pages, Code_publish)
    VALUES (nextCode, pTitle, pAuthorCode, pPages, pPublishCode);

    SELECT * FROM books WHERE Code_book = nextCode;
END //
DELIMITER ;

```

Процедура добавляет новую книгу в базу данных. Автоматически генерирует новый id на 1 больше максимального. После вставки данных возвращает информацию о добавленной книге.

5. Определить поставки с минимальной и максимальной стоимостью книг. Отобразить список всех поставок. Если стоимость поставки – максимальная, то вывести сообщение «Максимальная стоимость», если стоимость – минимальная, то вывести сообщение «Минимальная стоимость», иначе – «Средняя стоимость».

```

DELIMITER //

CREATE PROCEDURE AnalyzePurchaseCosts()
BEGIN
    DECLARE minCost FLOAT;
    DECLARE maxCost FLOAT;

    -- Находим минимальную и максимальную общую стоимость покупок
    SELECT MIN(Cost * Amount), MAX(Cost * Amount)
    INTO minCost, maxCost
    FROM purchases;

    -- Анализируем каждую покупку и определяем ее статус стоимости
    SELECT
        p.*,
        b.Title_book,
        (p.Cost * p.Amount) AS TotalCost,
        CASE
            WHEN (p.Cost * p.Amount) = minCost THEN 'Минимальная стоимость'
            WHEN (p.Cost * p.Amount) = maxCost THEN 'Максимальная стоимость'
            ELSE 'Средняя стоимость'
        END AS CostStatus
    FROM purchases p
    JOIN books b ON p.Code_book = b.Code_book;

END //

```

Процедура анализирует стоимость всех покупок книг. Сначала определяет минимальную и максимальную общую стоимость (цена * количество). Затем для каждой покупки выводит информацию о книге, общую стоимость и статус

Code_purchase	Code_book	Date_order	Code_delivery	Type_purchase	Cost	Amount	Title_book	TotalCost	CostStatus
3	4	2024-03-26	1	retail	113.172	60	Pride and Prejudice	6790.31982421875	Средняя стоимость
6	6	2024-08-19	8	opt	66.36	10	War and Peace	663.6000061035156	Средняя стоимость
7	9	2023-12-01	9	retail	32.652	19	Crime and Punishment	620.3880081176758	Средняя стоимость
14	6	2017-06-28	2	opt	94.848	110	War and Peace	10433.27995300293	Средняя стоимость
24	9	2024-06-19	8	opt	117.732	80	Crime and Punishment	9418.560180664062	Средняя стоимость
27	8	2014-02-11	2	retail	53.724	10	The Odyssey	537.239990234375	Средняя стоимость
38	2	2024-06-12	8	opt	106.584	30	To Kill a Mockingbird	3197.519989013672	Средняя стоимость
40	3	2015-10-16	3	retail	13.032	22	1984	286.7039909362793	Средняя стоимость
46	3	2024-07-03	6	opt	14.688	15	1984	220.31999588012695	Средняя стоимость
300	5	2023-12-26	1	opt	1200	6	Moby Dick	7200	Средняя стоимость

6. Определить количество записей в таблице поставщиков. Пока записей меньше 10, делать в цикле добавление записи в таблицу с автоматическим наращиванием значения ключевого поля, а вместо названия поставщика ставить значение 'не известен'.

```

DELIMITER //
CREATE PROCEDURE FillSuppliersToTen()
BEGIN
    DECLARE supplierCount INT;
    DECLARE nextCode INT;

    SELECT COUNT(*) INTO supplierCount FROM deliveries;

    WHILE supplierCount < 10 DO
        SELECT IFNULL(MAX(Code_delivery), 0) + 1 INTO nextCode FROM deliveries;

        INSERT INTO deliveries (Code_delivery, Name_delivery, Name_company, Address, Phone, INN)
        VALUES (nextCode, 'Не известен', 'Не известен', 'Не указан', 0, '000000000000');

        SET supplierCount = supplierCount + 1;
    END WHILE;
    SELECT * FROM deliveries;
END //
DELIMITER ;

```

Процедура проверяет количество записей в таблице поставщиков и, если их меньше 10, добавляет недостающих поставщиков с заполненными базовыми значениями. Каждый новый поставщик получает уникальный код. В конце процедура возвращает полный список поставщиков.

Code_delivery	Name_delivery	Name_company	Address	Phone	INN
1	Delivery Service 1	ОАО Филін	1 Main Street, City 1	1754652736	5314563286295
2	Delivery Service 2	Company 2	2 Main Street, City 2	1381159505	3192046376272
3	Delivery Service 3	Company 3	3 Main Street, City 3	2014584032	3151832425733
4	Delivery Service 4	Company 4	4 Main Street, City 4	9011375373	8308286089394
5	Delivery Service 5	ОАО Tower	5 Main Street, City 5	7012723100	9822222975685
6	Delivery Service 6	Company 6	6 Main Street, City 6	4523689063	3573031157685
7	Delivery Service 7	Company 7	7 Main Street, City 7	4242833963	
8	Delivery Service 8	Company 8	none info	7024707704	7047673976332
9	Delivery Service 9	ОАО KushTau	9 Main Street, City 9	3257448204	8404011550381
10	Delivery Service 10	Company 10	none info	4746057333	8962368079031

Хранимых процедур «Успеваемость».

1. Вывести фамилии и имена студентов (поля Surname, Name из таблицы Students) с максимальным средним баллом за весь период обучения (условие по полю Estimate из таблицы Progress).

```
CREATE OR REPLACE FUNCTION get_top_students()
RETURNS TABLE (surname VARCHAR(50), name VARCHAR(50), avg_estimate NUMERIC) AS $$
BEGIN
    RETURN QUERY
    SELECT s.surname, s.name, AVG(p.estimate)::NUMERIC(10,2) as avg_estimate
    FROM students s
    JOIN progress p ON s.code_stud = p.code_stud
    GROUP BY s.code_stud, s.surname, s.name
    HAVING AVG(p.estimate) = (
        SELECT MAX(avg_est)
        FROM (
            SELECT AVG(estimate) as avg_est
            FROM progress
            GROUP BY code_stud
        ) as max_avg
    );
END;
$$ LANGUAGE plpgsql;

SELECT * FROM get_top_students();
```

Процедура группирует данные по студентам и вычисляет средний балл. С помощью подзапроса определяется максимальный средний балл среди всех студентов, и возвращаются только тех, чей балл равен этому максимуму. Результат включает фамилию, имя и средний балл.

	surname character varying	name character varying	avg_estimate numeric
1	Чернов	Никита	4.67

2. Определить средний балл определенного студента (ФИО студента является входным параметром).

```
CREATE OR REPLACE FUNCTION get_student_avg_grade(student_name VARCHAR, student_surname VARCHAR)
RETURNS NUMERIC AS $$
DECLARE
    avg_grade NUMERIC;
BEGIN
    SELECT AVG(p.estimate)::NUMERIC(10,2) INTO avg_grade
    FROM students s
    JOIN progress p ON s.code_stud = p.code_stud
    WHERE s.name = student_name AND s.surname = student_surname;

    RETURN avg_grade;
END;
$$ LANGUAGE plpgsql;
```

Процедура вычисляет средний балл студента, соединяет таблицы students и progress, фильтрует записи по указанному студенту и возвращает среднее значение его оценок.

```
select * from get_student_avg_grade('Сафаргулов', 'Ильнар');
```

	get_student_avg_grade numeric
1	4.33

3. Определить специальность и номер курса определенного студента (ФИО студента является входным параметром, Название специальности и Номер курса – выходными параметрами).

```
CREATE OR REPLACE FUNCTION get_student_info(  
    IN student_name VARCHAR,  
    IN student_surname VARCHAR,  
    OUT speciality VARCHAR(100),  
    OUT course INTEGER  
) AS $$  
BEGIN  
    SELECT g.name_speciality, g.num_course INTO speciality, course  
    FROM students s  
    JOIN groups g ON s.code_group = g.code_group  
    WHERE s.name = student_name AND s.surname = student_surname;  
END;  
$$ LANGUAGE plpgsql;  
SELECT * FROM get_student_info('Алексей', 'Смирнов');
```

Процедура возвращает специальность и курс студента по его имени и фамилии, соединяет таблицы students и groups, находит указанного студента и извлекает данные о его группе.

```
select * from get_student_info('Сафаргулов', 'Ильнар')
```

	speciality character varying	course integer
1	Информатика и вычислительная техника	1

4. Выполните операцию вставки в таблицу Students. Код студента должен автоматически увеличиваться на единицу.

```
CREATE OR REPLACE FUNCTION insert_student(  
    p_surname VARCHAR(50),  
    p_name VARCHAR(50),  
    p_lastname VARCHAR(50),  
    p_birthday DATE,  
    p_phone NUMERIC,  
    p_code_group INTEGER  
) RETURNS VOID AS $$  
BEGIN  
    INSERT INTO students (surname, name, lastname, birthday, phone, code_group)  
    VALUES (p_surname, p_name, p_lastname, p_birthday, p_phone, p_code_group);  
END;  
$$ LANGUAGE plpgsql;  
  
SELECT insert_student('Петров', 'Сергей', 'Иванович', '2001-02-15', 79161234567, 1);
```

Процедура принимает все необходимые данные и выполняет вставку.

```
select * from public.insert_student('Дехканова', 'Джамиля', '--', '2006-12-14', '89871891795', 1);  
select * from students
```

tud eger	surname character varying (50)	name character varying (50)	lastname character varying (50)	birthday date	phone numeric	code_group integer	avg_estimate numeric (5,2)
5	Илиязетдинов	Роберт	Рушанович	2006-10-03	89625431505	1	0.00
6	Чернов	Никита	Димович	2006-12-12	89191901795	2	0.00
11	Иванов	Павел	Сергеевич	2006-12-21	89196132234	2	0.00
12	Дехканова	Джамиля	--	2006-12-14	89871891795	1	0.00

5. Определить средний возраст всех студентов. Вывести список всех студентов. Если возраст студента больше среднего возраста, то вывести сообщение «Вы старше среднего возраста всех студентов», если возраст – меньше, то вывести сообщение «Ваш возраст меньше среднего возраста всех студентов», а иначе – «Ваш возраст равен среднему возрасту всех студентов».


```

CREATE OR REPLACE FUNCTION get_students_with_age_comparison()
RETURNS TABLE (
    surname VARCHAR(50),
    name VARCHAR(50),
    age INTEGER,
    age_comparison TEXT
) AS $$
DECLARE
    avg_age NUMERIC;
BEGIN
    -- Вычисляем средний возраст
    SELECT AVG(EXTRACT(YEAR FROM AGE(birthday))) INTO avg_age
    FROM students;

    -- Возвращаем студентов с сравнением возраста
    RETURN QUERY
    SELECT
        s.surname,|
        s.name,
        EXTRACT(YEAR FROM AGE(s.birthday))::INTEGER as age,
        CASE
            WHEN EXTRACT(YEAR FROM AGE(s.birthday)) > avg_age THEN 'Вы старше среднего возраста'
            WHEN EXTRACT(YEAR FROM AGE(s.birthday)) < avg_age THEN 'Ваш возраст меньше среднего'
            ELSE 'Ваш возраст равен среднему возрасту всех студентов'
        END as age_comparison
    FROM students s;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM get_students_with_age_comparison();

```

Процедура сравнивает возраст каждого студента со средним возрастом по группе. Сначала вычисляется средний возраст всех студентов, затем для каждого возвращается фамилия, имя, возраст и текстовое сравнение.

```
select * from get_students_with_age_comparison()
```

Output Messages Notifications

surname character varying	name character varying	age integer	age_comparison text
Сафаргулов	Ильнар	19	Вы старше среднего возраста
Феденев	Владислав	18	Вы старше среднего возраста
Бахонов	Эмир	18	Вы старше среднего возраста
Давлатов	Минир	16	Ваш возраст меньше среднего
Гилязетдинов	Роберт	18	Вы старше среднего возраста
Чернов	Никита	18	Вы старше среднего возраста
Иванов	Павел	18	Вы старше среднего возраста
Дехканова	Джамиля	18	Вы старше среднего возраста

6. Определить количество записей в таблице дисциплин. Пока записей меньше 10, делать в цикле добавление записи в таблицу с автоматическим наращиванием значения ключевого поля, а вместо названия дисциплины ставить значение 'не известно'.

```

CREATE OR REPLACE FUNCTION fill_subjects_to_ten()
RETURNS VOID AS $$
DECLARE
    subject_count INTEGER;
BEGIN
    -- Получаем текущее количество дисциплин
    SELECT COUNT(*) INTO subject_count FROM subjects;

    -- Добавляем записи, пока их меньше 10
    WHILE subject_count < 10 LOOP
        INSERT INTO subjects (name_subject, count_hours)
        VALUES ('не известно', 0);

        subject_count := subject_count + 1;
    END LOOP;
END;
$$ LANGUAGE plpgsql;

SELECT fill_subjects_to_ten();

```

Процедура дополняет таблицу предметов до 10 записей, если их меньше. Каждая новая запись получает название "не известно" и 0 часов. Использует цикл для добавления недостающих записей.

code_subject [PK] integer	name_subject character varying (100)	count_hours integer
1	Математика	98
2	Физика	56
3	не известно	0
4	не известно	0
5	не известно	0
6	не известно	0
7	не известно	0
8	не известно	0
9	не известно	0
10	не известно	0

Триггеры

Триггеры MySql

1. Создайте триггер, запускаемый при занесении новой строки в таблицу Авторы. Триггер должен увеличивать счетчик числа добавленных строк.

```
DELIMITER //
CREATE TRIGGER after_author_insert
AFTER INSERT ON authors
FOR EACH ROW
BEGIN
    SET @author_count = @author_count + 1;
END//
DELIMITER ;
```

Триггер срабатывает после каждой вставки новой записи в таблицу .
Увеличивает значение счетчика на 1.

2. Добавьте в таблицу Авторы поле Количество книг (Count_books) целого типа со значением по умолчанию 0. Создайте хранимую процедуру, которая подсчитывает количество книг по каждому автору и заносит в поле Count_books эту информацию. Создайте триггер, запускаемый после внесения новой информации о книге.

```
ALTER TABLE `authors` ADD COLUMN Count_books INT DEFAULT 0;

DELIMITER //
CREATE PROCEDURE update_author_book_count()
BEGIN
    UPDATE authors a
    SET a.Count_books = (
        SELECT COUNT(*)
        FROM books b
        WHERE b.Code_author = a.Code_author
    );
END//
DELIMITER ;
```

Добавление нового столбца и процедура которая подсчитывает кол-во книг у авторов.

```
DELIMITER //
CREATE TRIGGER after_book_insert
AFTER INSERT ON books
FOR EACH ROW
> BEGIN
    UPDATE authors
    SET Count_books = Count_books + 1
    WHERE Code_author = NEW.Code_author;
~ END//
DELIMITER ;
```

Триггер автоматически обновляет счетчик при добавлении новой книги.

3. Создайте триггер, запускаемый при внесении информации о новых поставках. Выполните проверку о количестве добавляемой книги в таблице Книги. Если количество экземпляров книг в таблице меньше 10, то необходимо увеличить стоимость книг на 20 %.

```
DELIMITER //
CREATE TRIGGER after_purchase_insert
AFTER INSERT ON purchases
FOR EACH ROW
BEGIN
    DECLARE book_quantity INT;
    DECLARE book_title VARCHAR(255);

    SELECT SUM(Amount) INTO book_quantity
    FROM purchases
    WHERE Code_book = NEW.Code_book;

    SELECT Title_book INTO book_title FROM books WHERE Code_book = NEW.Code_book;
```

```

IF book_quantity < 10 THEN
    UPDATE purchases
    SET Cost = Cost * 1.2
    WHERE Code_book = NEW.Code_book;

    INSERT INTO price_change_logs (book_id, book_title, old_price, new_price, change_date)
    VALUES (NEW.Code_book, book_title, NEW.Cost, NEW.Cost*1.2, NOW());
END IF;
END//
DELIMITER ;

```

Триггер проверяет количество книг после каждой новой покупки. Если остаток меньше 10, цена увеличивается на 20%.

4. Запретить вставлять новые строки в таблицу Поставщики, выводя при этом сообщение «Вставка строк запрещена».

```

DELIMITER //
CREATE TRIGGER before_delivery_insert
BEFORE INSERT ON deliveries
FOR EACH ROW
BEGIN
    SIGNAL SQLSTATE '45000'
    SET MESSAGE_TEXT = 'Вставка строк запрещена ';
END//
DELIMITER ;

```

Триггер полностью блокирует добавление новых записей в таблицу "Поставщики", выводя сообщение об ошибке

5. Проверьте выполнение команд транзакции при добавлении новой информации об издательствах.


```

DELIMITER //
CREATE PROCEDURE add_publisher(
    IN p_code INT,
    IN p_name VARCHAR(100),
    IN p_city VARCHAR(50)
)
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
        SELECT 'Ошибка добавлении' AS error;
    END;

    START TRANSACTION;

    INSERT INTO publishing_house (Code_publish, Publish, City)
    VALUES (p_code, p_name, p_city);

    COMMIT;
    SELECT CONCAT('Издательство "', p_name, '" добавлено') AS result;

```

Триггер позволяет безопасно добавлять новые издательства с возможностью отката изменений в случае ошибки.

Триггеры postgresql

1. Создайте триггер, запускаемый при занесении новой строки в таблицу Преподаватели. Триггер должен увеличивать счетчик числа добавленных строк.

```

CREATE OR REPLACE FUNCTION increment_lector_count()
RETURNS TRIGGER AS $$
BEGIN

    IF NOT EXISTS (SELECT 1 FROM pg_tables WHERE tablename = 'counters') THEN
        CREATE TABLE counters (
            counter_id SERIAL PRIMARY KEY,
            lector_count INT DEFAULT 0
        );
        INSERT INTO counters (counter_id, lector_count) VALUES (1, 0);
    END IF;

    UPDATE counters SET lector_count = lector_count + 1 WHERE counter_id = 1;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER lector_insert_trigger
AFTER INSERT ON lectors
FOR EACH ROW
EXECUTE FUNCTION increment_lector_count();

```

Триггер автоматически увеличивает счетчик в таблице. при каждом добавлении нового преподавателя. Если таблица не существует, он создает ее перед обновлением.

2. Добавьте в таблицу Студенты поле Средний балл (Avg_Estimate) вещественного типа со значением по умолчанию 0. Создайте хранимую процедуру, которая подсчитывает средний балл для каждого студента и заносит в поле Avg_Estimate эту информацию. Создайте триггер, запускаемый после внесения новой информации об оценках студента и автоматически обновляет информацию о среднем балле студента.

```
ALTER TABLE students ADD COLUMN avg_estimate NUMERIC(5,2) DEFAULT 0.0;
```

```
CREATE OR REPLACE FUNCTION update_all_avg_estimates()  
RETURNS TRIGGER AS $$  
BEGIN  
    UPDATE students s  
    SET avg_estimate = COALESCE((  
        SELECT AVG(estimate)  
        FROM progress p  
        WHERE p.code_stud = s.code_stud  
    ), 0.0);  
  
    RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER update_avg_trigger  
AFTER INSERT OR UPDATE OR DELETE ON progress  
FOR EACH STATEMENT  
EXECUTE FUNCTION update_all_avg_estimates();
```

Триггер после любых изменений в таблице успеваемости , пересчитывает средний балл для всех затронутых студентов. Используется COALESCE для обработки нулевых значений.

3. Создайте триггер, запускаемый при внесении информации о новых оценках. Выполните проверку наличия информации о добавляемом студенте в таблице Студенты. Если данная информация в таблице отсутствует, то необходимо запустить хранимую процедуру на вставку записи в таблицу Студенты (параметры можно задать произвольно).

```
CREATE OR REPLACE FUNCTION insert_missing_student()
RETURNS TRIGGER AS $$
BEGIN
    IF NOT EXISTS (SELECT 1 FROM students WHERE code_stud = NEW.code_stud) THEN
        INSERT INTO students (
            code_stud,
            surname,
            name,
            birthday,
            code_group
        )
        VALUES (
            NEW.code_stud,
            'Бекмухамедов ',
            'Салават',
            '2005-01-01',
            1
        );
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER check_student_before_insert
BEFORE INSERT ON progress
FOR EACH ROW
EXECUTE FUNCTION insert_missing_student();
```

Триггер перед добавлением оценки проверяет существование студента. Если студент не найден, создается новая запись с данными.

4. Запретить вставлять новые строки в таблицу Группы, выводя при этом сообщение «Вставка строк запрещена».

```
CREATE OR REPLACE FUNCTION prevent_insert()  
RETURNS TRIGGER AS $$  
BEGIN  
    RAISE EXCEPTION 'Вставка строк запрещена';  
END;  
$$ LANGUAGE plpgsql;  
  
CREATE TRIGGER no_insert_groups  
BEFORE INSERT ON groups  
FOR EACH ROW  
EXECUTE FUNCTION prevent_insert();
```

Триггер блокирует добавление новых групп через исключение с сообщением.

5. Проверьте выполнение команд транзакции при добавлении новой информации о преподавателях.

```

CREATE OR REPLACE FUNCTION add_lector(
    p_name_lector VARCHAR(100),
    p_science VARCHAR(100),
    p_post VARCHAR(50)
)
RETURNS TEXT AS $$
DECLARE
    result_text TEXT;
BEGIN

    INSERT INTO lectors (name_lector, science, post, date_)
    VALUES (p_name_lector, p_science, p_post, CURRENT_DATE);

    IF NOT FOUND THEN
        RAISE EXCEPTION 'Не удалось добавить преподавателя';
    END IF;

    RETURN 'Преподаватель ' || p_name_lector || ' добавлен';

EXCEPTION
    WHEN OTHERS THEN

        RETURN 'Ошибка при добавлении : ' || SQLERRM;
END;
$$ LANGUAGE plpgsql;

```

Функция добавление преподавателя с обработкой ошибок. При успехе возвращает подтверждение, при ошибке проблему.

Пользователи базы данных «Книжное дело» в MySQL.

1. Администратор – обладает всеми правами

- `START TRANSACTION;`
- `CREATE USER 'admin_bookbiz'@'localhost' IDENTIFIED BY 'Admin@1234';`
- `GRANT ALL PRIVILEGES ON book_business_bd.* TO 'admin_bookbiz'@'localhost';`
- `GRANT PROXY ON ''@'' TO 'admin_bookbiz'@'localhost' WITH GRANT OPTION;`
- `COMMIT;`

```
GRANT USAGE ON *.* TO 'admin_bookbiz'@'localhost' WITH GRANT OPTION
GRANT ALL PRIVILEGES ON `book_business`.* TO 'admin_bookbiz'@'localhost'
GRANT PROXY ON ``@`` TO 'admin_bookbiz'@'localhost' WITH GRANT OPTION
```

2. Диспетчер – просматривает, заполняет и изменяет справочники: книги, авторы, издательства, поставщики.

```
START TRANSACTION;

CREATE USER 'dispatcher_book'@'localhost' IDENTIFIED BY 'Disp@2023';

GRANT SELECT, INSERT, UPDATE ON book_business_bd.authors TO 'dispatcher_book'@'localhost';
GRANT SELECT, INSERT, UPDATE ON book_business_bd.books TO 'dispatcher_book'@'localhost';
GRANT SELECT, INSERT, UPDATE ON book_business_bd.publishing_hou... TO 'dispatcher_book'@'localhost';
GRANT SELECT, INSERT, UPDATE ON book_business_bd.deliveries TO 'dispatcher_book'@'localhost';

COMMIT;
```

```
GRANT USAGE ON *.* TO 'dispatcher_book'@'localhost'
```

3. Менеджер по работе с поставщиками – просматривает и добавляет новую информацию в справочники, оформляет поставки.

```
START TRANSACTION;

CREATE USER 'supply_manager'@'localhost' IDENTIFIED BY 'Manager@456';

GRANT SELECT ON book_business_bd.authors TO 'supply_manager'@'localhost';
GRANT SELECT ON book_business_bd.books TO 'supply_manager'@'localhost';
GRANT SELECT ON book_business_bd.publishing_hou... TO 'supply_manager'@'localhost';
GRANT SELECT ON book_business_bd.deliveries TO 'supply_manager'@'localhost';

GRANT INSERT ON book_business_bd.deliveries TO 'supply_manager'@'localhost';

GRANT SELECT, INSERT ON book_business_bd.purchases TO 'supply_manager'@'localhost';

COMMIT;
```

Grants for supply_manager@localhost
GRANT USAGE ON *.* TO 'supply_manager'@'localhost'
GRANT SELECT ON `book_business_bd`.`books` TO 'supply_manager'@'localhost'
GRANT INSERT ON `book_business_bd`.`deliveries` TO 'supply_manager'@'localhost'
GRANT SELECT, INSERT ON `book_business_bd`.`purchases` TO 'supply_manager'@'localhost'

4. Поставщики – просматривают только свои поставки

```
START TRANSACTION;
```

```
CREATE VIEW supplier1_purchases AS
```

```
SELECT p.*
```

```
FROM purchases p
```

```
JOIN deliveries d ON p.Code_delivery = d.Code_delivery
```

```
WHERE d.Code_delivery = 1;
```

```
CREATE USER 'supplier1'@'localhost' IDENTIFIED BY 'Supplier1@Pass';
```

```
GRANT SELECT ON book_business_bd.supplier1_purchases TO 'supplier1'@'localhost';
```

```
GRANT SELECT ON book_business_bd.supplier1_purchases TO 'supplier1'@'localhost';
```

```
CREATE VIEW supplier2_purchases AS
```

```
SELECT p.*
```

```
FROM purchases p
```

```
JOIN deliveries d ON p.Code_delivery = d.Code_delivery
```

```
WHERE d.Code_delivery = 2;
```

```
CREATE USER 'supplier2'@'localhost' IDENTIFIED BY 'Supplier2@Pass';
```

```
GRANT SELECT ON book_business_bd.supplier2_purchases TO 'supplier2'@'localhost';
```

```
COMMIT;
```

Grants for supplier1@localhost
▶ GRANT USAGE ON *.* TO 'supplier1'@'localhost'
GRANT SELECT ON `book_business_bd`.`supplier1_purchases` TO 'supplier1'@'localhost'

GRANT USAGE ON *.* TO 'supplier2'@'localhost'
GRANT SELECT ON `book_business_bd`.`supplier2_purchases` TO 'supplier2'@'localhost'