МОДУЛЬ SQL

/*Запросы

1.Покажите среднюю зарплату сотрудников за каждый год (средняя заработная плата среди тех, кто работал в отчетный период -статистика с начала до 2005 года).*/

```
SELECT
    YEAR(from_date) AS 'YEAR',
    ROUND(AVG(salary), 2) AS AVERGE_SALARY
FROM
    salaries
WHERE
    YEAR(from_date) <= 2005
GROUP BY YEAR(from_date)
ORDER BY YEAR(from_date);
```

/*2.Покажите среднюю зарплату сотрудников по каждому отделу. Примечание: принять в расчет только текущие отделы и текущую заработную плату.*/

```
SELECT
    d.dept_name AS DEPARTMENT,
    ROUND(AVG(salary), 2) AS AVERGE_SALARY
FROM
    salaries AS s
        INNER JOIN
    dept_emp AS de USING (emp_no)
        INNER JOIN
    departments AS d ON (de.dept_no = d.dept_no)
WHERE
    CURDATE() BETWEEN de.from_date AND de.to_date
GROUP BY d.dept_name
ORDER BY d.dept_name;
```

/*3.Покажите среднюю зарплату сотрудников по каждому отделу за каждый год. Примечание: для средней зарплаты отдела Xв году Y нам нужно взять среднее значение всех зарплат в году Yсотрудников,которые были в отделе Xв году Y.*/

```
SELECT
    YEAR(s.from_date) AS 'YEAR',
    d.dept_name AS 'DEPARTMENT',
    ROUND(AVG(s.salary), 2) 'AVG_SALARY'
FROM
    salaries AS s
        INNER JOIN
    dept_emp AS de USING (emp_no)
        INNER JOIN
    departments AS d ON (de.dept_no = d.dept_no)
WHERE
    s.from_date = de.from_date
GROUP BY YEAR(s.from_date) , d.dept_name
ORDER BY YEAR(s.from_date);
```

/*4.Покажите для каждого года самый крупный отдел (по количеству сотрудников) в этом году и его среднюю зарплату.*/

```
CREATE VIEW tab1 AS
    (SELECT
        YEAR(s.from_date) AS YEAR_OF_WORK,
        d.dept_name AS DEPT,
        COUNT(s.emp_no) AS EMP_NUMBER,
        ROUND(AVG(s.salary), 2) AS AVG_SALARY
    FROM
        salaries AS s
            INNER JOIN
        dept_emp AS de USING (emp_no)
            INNER JOIN
        departments AS d ON (de.dept_no = d.dept_no)
    WHERE
        s.from_date = de.from_date
    GROUP BY YEAR(s.from_date) , d.dept_no
    ORDER BY YEAR(s.from_date));

SELECT
    tab1.YEAR_OF_WORK,
    tab1.DEPT,
    tab1.EMP_NUMBER,
    tab1.AVG_SALARY
FROM
    tab1
        INNER JOIN
    (SELECT
        MAX(EMP_NUMBER) AS MAX_COUNT
    FROM
        tab1
    GROUP BY YEAR_OF_WORK) AS tab2 ON (tab1.EMP_NUMBER = tab2.MAX_COUNT);

DROP VIEW tab1;
```

/*5.Покажите подробную информацию о менеджере, который дольше всех исполняет свои обязанности на данный момент.*/

```
SELECT
    CONCAT(e.first_name, '  ', e.last_name) AS FULL_NAME_MANAGER,
    e.birth_date AS BIRTH_DAY,
    e.hire_date AS HIRE_DATE,
    d.dept_name AS DEPARTMENT,
    t.title AS TITLE,
    DATEDIFF(CURDATE(), dm.from_date) AS DAYS_OF_WORKING_AS_MANAGER
FROM
    dept_manager AS dm
        INNER JOIN
    employees AS e USING (emp_no)
        INNER JOIN
    titles AS t USING (emp_no)
        INNER JOIN
    departments AS d ON (dm.dept_no = d.dept_no)
WHERE
    (CURDATE() BETWEEN dm.from_date AND dm.to_date)
        AND dm.from_date = t.from_date
        AND DATEDIFF(CURDATE(), dm.from_date) =
                                        (SELECT
                                                MAX(DATEDIFF(CURDATE(), from_date))
                                        FROM
                                                dept_manager
                                        WHERE
                                                CURDATE() BETWEEN from_date AND to_date
                                        );
```

/*6.Покажите топ-10 нынешних сотрудников компании с наибольшей разницей между их зарплатой и текущей средней зарплатой в их отделе.*/

```
WITH tab AS
        (
        SELECT
                de.dept_no,
                AVG(s.salary) AS avg_sal
        FROM
                salaries AS s
                        INNER JOIN
                dept_emp AS de USING (emp_no)
        WHERE
                CURDATE() BETWEEN de.from_date AND de.to_date AND
      CURDATE() BETWEEN s.from_date AND s.to_date
        GROUP BY de.dept_no
        )
SELECT
   CONCAT(e.first_name, '  ', e.last_name) AS FULL_NAME,
   d.dept_name AS DEPARTMENT,
   s.salary- avg_sal AS DIFF_SALARY_and_AVG_SALARY
FROM
                employees AS e
        INNER JOIN
                salaries AS s USING (emp_no)
        INNER JOIN
                dept_emp AS de USING (emp_no)
        INNER JOIN
                departments AS d ON (de.dept_no=d.dept_no)
        INNER JOIN
                tab ON (tab.dept_no=de.dept_no)
        WHERE
                CURDATE() BETWEEN de.from_date AND de.to_date AND
      CURDATE() BETWEEN s.from_date AND s.to_date
ORDER BY s.salary- avg_sal DESC
LIMIT 10;
```

/*7.Из-за кризиса на одно подразделение на своевременную выплату зарплаты выделяется всего 500 тысяч долларов. Правление решило, что низкооплачиваемые сотрудники будут первыми получать зарплату. Показать список всех сотрудников, которые будут вовремя получать зарплату (обратите внимание, что мы должны платить зарплату за один месяц, но в базе данных мы храним годовые суммы).*/

```
WITH tab AS
	(
  SELECT
		e.first_name, e.last_name,
		d.dept_name,
		SUM(s.salary/12) OVER
		(PARTITION BY de.dept_no ORDER BY s.salary ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
			AS total
	FROM
		employees AS e
			INNER JOIN
		salaries AS s USING (emp_no)
			INNER JOIN
		dept_emp AS de USING (emp_no)
			INNER JOIN
		departments AS d ON (de.dept_no=d.dept_no)
	WHERE
		CURDATE() BETWEEN de.from_date AND de.to_date
			AND CURDATE() BETWEEN s.from_date AND s.to_date
	ORDER BY de.dept_no , s.salary
  )
SELECT
	ROW_NUMBER() OVER (PARTITION BY tab.dept_name) AS NUM,
	tab.dept_name AS DEPARTMENT,
	CONCAT(tab.first_name, '  ', tab.last_name) AS FULL_NAME
FROM
	tab
WHERE tab.total<=500000;
```

/*Дизайн базы данных:

1.Разработайте базу данных для управления курсами. База данных содержит следующие сущности: a.students: student_no, teacher_no, course_no, student_name, email, birth_date.b.teachers: teacher_no, teacher_name, phone_no c.courses: course_no, course_name, start_date, end_date. Секционировать по годам, таблицу students по полю birth_date с помощью механизма range. В таблице students сделать первичный ключ в сочетании двух полей student_no и birth_date. Создать индекс по полю students.email Создать уникальный индекс по полю teachers.phone_no */

```
CREATE DATABASE IF NOT EXISTS coursedb;
USE coursedb;
CREATE TABLE IF NOT EXISTS students
        (
   student_no INT UNSIGNED,
   teacher_no SMALLINT UNSIGNED,
   course_no SMALLINT UNSIGNED,
   student_name CHAR (50),
   email CHAR(30),
   birth_date DATE,
   INDEX(email),
   CONSTRAINT student PRIMARY KEY (student_no, birth_date)
   )
   ENGINE InnoDB
   PARTITION BY RANGE (year(birth_date))
                                        (PARTITION p19 VALUES LESS THAN (2000),
                                        PARTITION p20 VALUES LESS THAN (MAXVALUE));

CREATE TABLE IF NOT EXISTS teachers
        (
   teacher_no SMALLINT UNSIGNED,
   teacher_name CHAR (50),
   phone_no CHAR(13) UNIQUE
   )
   ENGINE InnoDB;

CREATE TABLE IF NOT EXISTS courses
        (
   course_no SMALLINT UNSIGNED,
   start_date DATE,
```

```
end_date DATE
)
ENGINE InnoDB;
```

```
/*2. На свое усмотрение добавить тестовые данные (7-10 строк) в наши три таблицы.*/
INSERT INTO students VALUES
(1,1,1,'Klymash Andrii','klymash@ukr.net','1978-07-22'),
(2,2,2,'Balkovska Ganna','balkovska@gmail.com','1985-05-31'),
(3,2,3,'Serhienko Oksana','seghienko@i.ua','1976-08-12'),
(4,4,4,'Tatarcheno Halina','tatarchenko@gmail.com','1965-10-4'),
(5,5,5,'Medvid Ivan','medvid@ukr.net','2001-12-05'),
(6,6,6,'Kortieva Olena','kortieva@gmail.com','2002-07-22'),
(7,7,7,'Pavlenko Svitlana','pavlenko@gmail.com','2005-08-24'),
(8,8,8,'Keremet Mykhailo','keremet@i.ua','2000-01-20');

INSERT INTO teachers VALUES
(1,'Zverkin Andrii','+380506243234'),
(2,'Kichkina Olena','+380504556235'),
(3,'Boyko Grygorii','+380665234578'),
(4,'Rozmyslov Oleksandr','+380506240208'),
(5,'Galgash Ruslan','+380506547896'),
(6,'Arsentieva Olena','+380991234558'),
(7,'Cherednychenko Serhii','+380502584758'),
(8,'Kalashnykov Igor','+380506321245');

INSERT INTO courses VALUES
(1,'2022-05-01','2022-05-31'),
(2,'2022-06-01','2022-06-30'),
(3,'2022-07-01','2022-07-31'),
(4,'2022-08-01','2022-08-31'),
(5,'2022-09-01','2022-09-30'),
(6,'2022-10-01','22-10-31'),
(7,'2022-11-01','2022-11-30'),
(8,'2022-12-01','2022-12-31');
```

/*3.Отобразить данные за любой год из таблицы students и зафиксировать в виду комментария план выполнения запроса, где будет видно что запрос будет выполняться по конкретной секции.*/

```
select * FROM students PARTITION (p19);
```

```
/*student_no, teacher_no, course_no, student_name,        email,                birth_date
 '1',       '1',       '1',       'Klymash Andrii',     'klymash@ukr.net',       '1978-07-22'
 '2',       '2',       '2',       'Balkovska Ganna',    'balkovska@gmail.com',   '1985-05-31'
 '3',       '2',       '3',       'Serhienko Oksana',   'seghienko@i.ua',        '1976-08-12'
 '4',       '4',       '4',       'Tatarcheno Halina',  'tatarchenko@gmail.com', '1965-10-04'*/
```

```
EXPLAIN SELECT * FROM students PARTITION (p19);
```

```
/* id,   select_type, table,     partitions, type,  possible_keys, key,  key_len, ref,  rows, filtered, Extra
  '1', 'SIMPLE',    'students', 'p19',      'ALL', NULL,          NULL, NULL,    NULL, '4', '100.00',  NULL
```

/*4.Отобразить данные учителя, по любому одному номеру телефона и зафиксировать план выполнения запроса, где будет видно, что запрос будет выполняться по индексу, а не методом ALL. Далее индекс из поля teachers.phone_no сделать невидимым и зафиксировать план выполнения запроса, где ожидаемый результат -метод ALL. В итоге индекс оставить в статусе -видимый. */

```
SELECT * FROM teachers WHERE phone_no='+380502584758';

/* teacher_no, teacher_name,            phone_no
   '7',        'Cherednychenko Serhii', '+380502584758'*/

EXPLAIN SELECT * FROM teachers WHERE phone_no='+380502584758';

/* with index 'phone_no'
   id,   select_type,  table,     partitions,  type,    possible_keys, key,       key_len, ref,    rows, filtered, Extra
  '1',   'SIMPLE',    'teachers', NULL,        'const',  'phone_no',   'phone_no', '53',   'const', '1', '100.00',  NULL

   without index 'phone_no'

 id,  select_type, table,     partitions, type, possible_keys, key,  key_len, ref,  rows, filtered, Extra
'1', 'SIMPLE',    'teachers', NULL,       'ALL', NULL,          NULL, NULL,    NULL, '8',  '12.50', 'Using where'*/
```

```
/*5.Специально сделаем 3 дубляжа в таблице students (добавим еще 3 одинаковые строки). */

ALTER TABLE students DROP PRIMARY KEY;

INSERT INTO students VALUES
(1,1,1,'Klymash Andrii','klymash@ukr.net','1978-07-22'),
(2,2,2,'Balkovska Ganna','balkovska@gmail.com','1985-05-31'),
(3,2,3,'Serhienko Oksana','seghienko@i.ua','1976-08-12');

SELECT
    *
FROM
    students;

/* student_no, teacher_no, course_no, student_name,         email,                 birth_date
   '1',        '1',        '1',       'Klymash Andrii',    'klymash@ukr.net',       '1978-07-22'
   '1',        '1',        '1',       'Klymash Andrii',    'klymash@ukr.net',       '1978-07-22'
   '2',        '2',        '2',       'Balkovska Ganna',   'balkovska@gmail.com',   '1985-05-31'
   '2',        '2',        '2',       'Balkovska Ganna',   'balkovska@gmail.com',   '1985-05-31'
   '3',        '2',        '3',       'Serhienko Oksana',  'seghienko@i.ua',        '1976-08-12'
   '3',        '2',        '3',       'Serhienko Oksana',  'seghienko@i.ua',        '1976-08-12'
   '4',        '4',        '4',       'Tatarcheno Halina', 'tatarchenko@gmail.com', '1965-10-04'
   '5',        '5',        '5',       'Medvid Ivan',       'medvid@ukr.net',        '2001-12-05'
   '6',        '6',        '6',       'Kortieva Olena',    'kortieva@gmail.com',    '2002-07-22'
   '7',        '7',        '7',       'Pavlenko Svitlana', 'pavlenko@gmail.com',    '2005-08-24'
   '8',        '8',        '8',       'Keremet Mykhailo',  'keremet@i.ua',          '2000-01-20'*/
```

```
/*6.Написать запрос, который выводит строки с дубляжами.*/

SELECT
   *
FROM
   students
GROUP BY student_name, student_no, teacher_no, course_no, email, birth_date
HAVING COUNT(student_name) > 1;

/* student_no, teacher_no, course_no, student_name,         email,                birth_date
  '1',        '1',        '1',       'Klymash Andrii',    'klymash@ukr.net',       '1978-07-22'
  '2',        '2',        '2',       'Balkovska Ganna',   'balkovska@gmail.com',   '1985-05-31'
  '3',        '2',        '3',       'Serhienko Oksana',  'seghienko@i.ua',        '1976-08-12'
```

МОДУЛЬ ETL

/* 1. Создать процедуру добавления нового сотрудника, с нужным перечнем входящий параметров. После успешной работы процедуры данные должны попасть в таблицы employees, dept_emp, salaries и titles; Вычисление emp_no, вычисляем по формуле max(emp_no)+1. Если передана не существующая должность, тогда показать ошибку с нужным текстом. Если передана зарплата меньше 30000, тогда показать ошибку с нужным текстом.*/

```
USE employees;
DELIMITER //
CREATE PROCEDURE new_employee (
b_d date,
f_n VARCHAR(50),
l_n VARCHAR(50),
gen CHAR(1),
dep VARCHAR(50),
tit VARCHAR(50),
sal INT
)
BEGIN
IF dep NOT IN (SELECT dept_name FROM departments)
THEN
signal sqlstate '45000'
set message_text = 'DEPARTMENT DOES NOT EXIST';
ELSEIF tit NOT IN (SELECT title FROM titles)
THEN
signal sqlstate '45000'
set message_text = 'TITLE DOES NOT EXIST';
ELSEIF sal < 30000
THEN
signal sqlstate '45000'
set message_text = 'LOW SALARY'
END IF;
SET @e_n= (select max(emp_no)+1 from employees);
INSERT INTO employees VALUES (@e_n, b_d, f_n, l_n, gen, curdate());
INSERT INTO dept_emp VALUES (@e_n,
(SELECT dept_no FROM
departments WHERE dept_name = dep),
```

```
curdate(),
'9999-01-01'
);
INSERT INTO salaries VALUES (@e_n,sal,curdate(),'9999-01-01');
INSERT INTO titles VALUES (@e_n,tit,curdate(),'9999-01-01');
END //
DELIMITER ;
START TRANSACTION;
CALL new_employee ('1973-09-24','Serhii','Kuzmenko','M','Sales','Staff',80000);
SELECT * FROM employees order by emp_no desc limit 1;
SELECT * FROM dept_emp order by emp_no desc limit 1;
SELECT * FROM salaries order by emp_no desc limit 1;
SELECT * FROM titles order by emp_no desc limit 1;
ROLLBACK;
COMMIT;
DROP PROCEDURE new_employee;
```

/* 2. Создать процедуру для обновления зарплаты по сотруднику. При обновлении зарплаты, нужно закрыть последнюю активную запись текущей датой, и создавать новую историческую запись текущей датой. Если передан ну существующий сотрудник, тогда показать ошибку с нужным текстом.*/

```
USE employees;

DELIMITER //
CREATE PROCEDURE new_salary (

e_n INT,

sal INT

)

BEGIN

IF e_n NOT IN (SELECT DISTINCT emp_no FROM employees)

THEN

signal sqlstate '45000'

set message_text = 'EMPLOYEE DOES NOT EXIST';

END IF;

UPDATE salaries SET to_date = curdate() WHERE emp_no=e_n AND to_date>curdate();

INSERT INTO salaries VALUES (e_n, sal, curdate(), '9999-01-01');

END //

DELIMITER ;

START TRANSACTION;

CALL new_salary (490000, 81000);

SELECT * FROM salaries WHERE emp_no=490000;

ROLLBACK;

COMMIT;

DROP PROCEDURE new_salary;
```

/*3. Создать процедуру для увольнения сотрудника, закрытия исторических записей в таблицах dept_emp, salaries и titles. Если передан несуществующий номер сотрудника, тогда показать ошибку с нужным текстом.*/

```
USE employees;
DELIMITER //
CREATE PROCEDURE dismissal_emloyee (e_n INT)
BEGIN
IF e_n NOT IN (SELECT DISTINCT emp_no FROM employees)
THEN
signal sqlstate '45000'
set message_text = 'EMPLOYEE DOES NOT EXIST';
ELSEIF curdate()>(SELECT max(to_date) FROM dept_emp WHERE emp_no=e_n)
THEN
signal sqlstate '45000'
set message_text = 'EMPLOYEE ALREADY FIRED';
END IF;
UPDATE dept_emp SET to_date = curdate() WHERE emp_no=e_n AND
to_date>curdate();
UPDATE salaries SET to_date = curdate() WHERE emp_no=e_n AND to_date>curdate();
UPDATE titles SET to_date = curdate() WHERE emp_no=e_n AND to_date>curdate();
END //
DELIMITER ;
START TRANSACTION;
CALL dismissal_emloyee (10020);
SELECT * FROM dept_emp WHERE emp_no=10020;
SELECT * FROM salaries WHERE emp_no=10020;
SELECT * FROM titles WHERE emp_no=10020;
ROLLBACK;
COMMIT;
DROP PROCEDURE dismissal_emloyee;
```

/*4. Создать функцию, которая выводила бы текущую зарплату по сотруднику.*/

```
USE employees;
DELIMITER //
CREATE FUNCTION current_salary (e_n INT)
RETURNS INT DETERMINISTIC
BEGIN
DECLARE cur_sal INT;
SET cur_sal = ( SELECT
salary
FROM
salaries
WHERE
emp_no = e_n AND to_date > CURDATE()
);
RETURN cur_sal;
END//
DELIMITER ;
SELECT current_salary (10009);
DROP FUNCTION current_salary;
```