Зачетная работа

Выполнил студент группы Б20-505

Сорочан Илья

Московский Инженерно-Физический Институт

Москва 2023

1 Задание

Задача данной работы: подробно описать процесс создания сложного SQL запроса к ранее разработанной в лабороторных базе данных.

Для каждого сотрудника вывести таблицу с:

1. именем;
2. названием должности;
3. общей суммой продаж (в которых он кассир);
4. число различных типов товаров, среди проданных им;
5. наименование самой распространенной категории товаров среди проданных им.

В следующей главе рассмотрим каждый пункт по порядку

2 Запросы

2.1 Имя и название должности

Первые два пункта получить относительно просто:

```
SELECT
  employee.name, job.name
FROM
  employee, job
WHERE
  employee.job = job.id;
```

Данный запрос выдаст:

```
Джон Стоун|Менеджер
Джек Воробей|Кассир
Партос|Уборщик
```

2.2 Общая сумма продаж

Здесь уже сложнее, так как требуется обработать данные из трех таблиц. В первую очередь нас интересует таблица sale, в которой хранятся все продажи. Из них мы создаем группы (окна) по продавцу. При это для суммирования цен нам понадобится таблица product, которая хранит айди покупки, к которой принадлежит. И наконец после получения результата его нужно будет объеденить с запросом из предыдущего подраздела.

Запрос:

```
SELECT
  employee.name, SUM(product.price)
FROM
  sale
```

```
JOIN
  employee ON sale.employee = employee.id
JOIN
  product ON sale.id = product.sale
GROUP BY
  employee.id;
```

Результат:

```
Джон Стоун|899
Джек Воробей|1299
Партос|129
```

Здесь мы сначала связываем в одну таблицу информацию о продажах, рабочих и товарах, а затем используем агрегатную функцию SUM с разделением по айди рабочих (основной ключ).

Теперь объединим данный запрос с запросом из предыдущей секции:

```
SELECT
  employee.name, job.name, SUM(product.price), COUNT(DISTINCT
product.type)
FROM
  sale
JOIN
  employee ON sale.employee = employee.id
JOIN
  product ON sale.id = product.sale
JOIN
  job ON employee.job = job.id
GROUP BY
  employee.id;
```

Выдает следующий результат:

```
Джон Стоун|Менеджер|899
```

```
Джек Воробей|Кассир|1299
Партос|Уборщик|129
```

2.3 Число различных типов товаров

Необходимо найти число различных типов товаров, проданных один продавцем. Данный запрос легко выполнить, добавив COUNT(DISTINCT product.type) в предыдущий запрос:

```
SELECT
  employee.name, job.name, SUM(product.price), COUNT(DISTINCT
product.type)
FROM
  sale
JOIN
  employee ON sale.employee = employee.id
JOIN
  product ON sale.id = product.sale
JOIN
  job ON employee.job = job.id
GROUP BY
  employee.id;
```

Результат:

```
Джон Стоун|Менеджер|899|1
Джек Воробей|Кассир|1299|1
Партос|Уборщик|129|1
```

2.4 Самая распространенная категория

Последний столбец конечного запроса должен содержать самую распространенную категорию, проданную рабочим.

Начнем с самого простого: получим таблицу и подсчетом всех категорий для всех продаж.

```
SELECT
  category.id, category.name, COUNT(*)
FROM
  category
JOIN
  product_type ON product_type.category = category.id
JOIN
  product ON product.type = product_type.code
WHERE
  product.sold = true
GROUP BY
  category.id;
```

Вывод:

```
1|Ручки|1
2|Тетради|1
3|Карандаши|1
```

Теперь, данную таблицу можно будет использовать в основном запросе, необходимо только добавить столбец с наименованием рабочего-кассира:

```
SELECT
  category.id, category.name, COUNT(*), sale.employee
FROM
  category
JOIN
  product_type ON product_type.category = category.id
JOIN
  product ON product.type = product_type.code
JOIN
  sale ON product.sale = sale.id
WHERE
  product.sold = true
GROUP BY
```

```
  category.id;
```

Выдает:

```
Ручки|1|1
Тетради|1|3
Карандаши|1|2
```

Наконец, объеденим данный запрос с предыдущим:

```
SELECT
  employee.name, job.name, SUM(product.price), COUNT(DISTINCT
product.type), category_count_list.category_name
FROM
  sale
JOIN
  employee ON sale.employee = employee.id
JOIN
  product ON sale.id = product.sale
JOIN
  job ON employee.job = job.id
JOIN
  (
    SELECT
      category.name as category_name, COUNT(*) AS category_count,
sale.employee
    FROM
      category
    JOIN
      product_type ON product_type.category = category.id
    JOIN
      product ON product.type = product_type.code
    JOIN
      sale ON product.sale = sale.id
    WHERE
      product.sold = true
```

```
    GROUP BY
      category.id, sale.employee
    HAVING
      category_count = (
        SELECT
          MAX(subquery.category_count)
        FROM
          (
            SELECT
              COUNT(*) AS category_count, sale.employee
            FROM
              category
            JOIN
              product_type ON product_type.category = category.id
            JOIN
              product ON product.type = product_type.code
            JOIN
              sale ON product.sale = sale.id
            WHERE
              product.sold = true
            GROUP BY
              category.id, sale.employee
          ) AS subquery
        WHERE
          subquery.employee = sale.employee
      )
  ) AS category_count_list ON category_count_list.employee =
employee.id
GROUP BY
  employee.id;
```

Здесь выполняется два подзапроса: один подсчитывает число категорий, другой отбирает максимальные.

3 Заключение

Задание было выполнено, запрос можно найти в последней секции предыдущей главы. От себя отмечу что коследний столбец был очень сложен, в силу чего я мог сделать неоптимизированный запрос.

Приложение А

Код для создания тестовой базы данных:

```
-- CREATE SECTION

CREATE TABLE vendor (
    id INTEGER PRIMARY KEY,
    name TEXT,
    office TEXT,
    rating INTEGER
);

CREATE TABLE category (
    id INTEGER PRIMARY KEY,
    name TEXT
);

CREATE TABLE product_type (
    code INTEGER PRIMARY KEY,
    vendor INTEGER,
    category INTEGER,
    raise INTEGER,
    FOREIGN KEY(vendor) REFERENCES vendor(id),
    FOREIGN KEY(category) REFERENCES category(id)
);

CREATE TABLE country (
    id INTEGER PRIMARY KEY,
    name TEXT
);

CREATE TABLE product (
    id INTEGER PRIMARY KEY,
```

```
    type INTEGER,
    country INTEGER,
    price INTEGER,
    supply INTEGER,
    sold BOOLEAN,
    sale INTEGER,
    FOREIGN KEY(type) REFERENCES product_type(code),
    FOREIGN KEY(country) REFERENCES country(id),
    FOREIGN KEY(supply) REFERENCES supply(id),
    FOREIGN KEY(sale) REFERENCES sale(id)
);

CREATE TABLE supply (
    id INTEGER PRIMARY KEY,
    date DATE,
    done BOOLEAN,
    employee INTEGER,
    price INTEGER,
    FOREIGN KEY(employee) REFERENCES employee(id)
);

CREATE TABLE employee (
    id INTEGER PRIMARY KEY,
    job INTEGER,
    name TEXT,
    active BOOLEAN,
    exp_date TIMESTAMP,
    FOREIGN KEY(job) REFERENCES job(id)
);

CREATE TABLE job (
    id INTEGER PRIMARY KEY,
    name TEXT,
    salary INTEGER,
    days_per_week INTEGER
```

```
);

CREATE TABLE sale (
    id INTEGER PRIMARY KEY,
    date TIMESTAMP,
    employee INTEGER,
    customer INTEGER,
    bonus INTEGER,
    bonus_used INTEGER,
    FOREIGN KEY(employee) REFERENCES employee(id),
    FOREIGN KEY(customer) REFERENCES customer(id)
);

CREATE TABLE customer (
    id INTEGER PRIMARY KEY,
    phone INTEGER,
    name TEXT,
    reg_date TIMESTAMP
);

-- INSERT SECTION

INSERT INTO vendor (id, name, office, rating) VALUES (1, 'Hatber',
'Россия, г. Москва, ул. X1, дом Y1', 8);
INSERT INTO vendor (id, name, office, rating) VALUES (2, 'Самсон',
'Россия, г. Воронеж, ул. X2, дом Y2', 7);
INSERT INTO vendor (id, name, office, rating) VALUES (3, 'Апплика',
'Россия, г. Москва, ул. X3, дом Y3', 6);

INSERT INTO category (id, name) VALUES (1, 'Ручки');
INSERT INTO category (id, name) VALUES (2, 'Тетради');
INSERT INTO category (id, name) VALUES (3, 'Карандаши');

INSERT INTO product_type (code, vendor, category, raise) VALUES
(1001, 1, 1, 15);
```

```
INSERT INTO product_type (code, vendor, category, raise) VALUES
(1002, 2, 1, 10);
INSERT INTO product_type (code, vendor, category, raise) VALUES
(1003, 1, 2, 20);
INSERT INTO product_type (code, vendor, category, raise) VALUES
(1004, 1, 3, 12);
INSERT INTO product_type (code, vendor, category, raise) VALUES
(1005, 3, 2, 19);

INSERT INTO country (id, name) VALUES (1, 'Германия');
INSERT INTO country (id, name) VALUES (2, 'Россия');
INSERT INTO country (id, name) VALUES (3, 'Китай');

INSERT INTO product (id, type, country, price, supply, sold,
sale) VALUES (1, 1001, 1, 999, 1, false, null);
INSERT INTO product (id, type, country, price, supply, sold,
sale) VALUES (2, 1002, 2, 899, 2, true, 1);
INSERT INTO product (id, type, country, price, supply, sold,
sale) VALUES (3, 1003, 3, 1499, 3, false, null);
INSERT INTO product (id, type, country, price, supply, sold,
sale) VALUES (4, 1004, 2, 1299, 2, true, 2);
INSERT INTO product (id, type, country, price, supply, sold,
sale) VALUES (5, 1005, 3, 129, 3, true, 3);

INSERT INTO supply (id, date, done, employee, price) VALUES (1,
'2023-06-01', true, 1, 850);
INSERT INTO supply (id, date, done, employee, price) VALUES (2,
'2023-06-15', true, 2, 750);
INSERT INTO supply (id, date, done, employee, price) VALUES (3,
'2023-06-30', false, 3, 1400);

INSERT INTO employee (id, job, name, active, exp_date) VALUES
(1, 1, 'Джон Стоун', true, '2023-12-31');
INSERT INTO employee (id, job, name, active, exp_date) VALUES
(2, 2, 'Джек Воробей', true, '2024-06-30');
```

```
INSERT INTO employee (id, job, name, active, exp_date) VALUES
(3, 3, 'Партос', false, '2023-09-30');

INSERT INTO job (id, name, salary, days_per_week) VALUES (1,
'Менеджер', 5000, 4);
INSERT INTO job (id, name, salary, days_per_week) VALUES (2,
'Кассир', 2500, 5);
INSERT INTO job (id, name, salary, days_per_week) VALUES (3,
'Уборщик', 500, 7);

INSERT INTO customer (id, phone, name, reg_date) VALUES (1, 1111111111,
'Вася Пупкин', '2022-01-15');
INSERT INTO customer (id, phone, name, reg_date) VALUES (2, 1111211111,
'Джек Ричер', '2022-02-28');
INSERT INTO customer (id, phone, name, reg_date) VALUES (3, 11111311111,
'Том Сойер', '2022-06-01');

INSERT INTO sale (id, date, employee, customer, bonus, bonus_used)
VALUES (1, '2023-06-20 10:30:00', 1, 1, 50, 0);
INSERT INTO sale (id, date, employee, customer, bonus, bonus_used)
VALUES (2, '2023-06-22 14:15:00', 2, 2, 25, 0);
INSERT INTO sale (id, date, employee, customer, bonus, bonus_used)
VALUES (3, '2023-06-23 11:00:00', 3, 3, 75, 75);
```