Санкт-Петербургский Политехнический Университет Петра Великого
Институт компьютерных наук и технологий
Кафедра компьютерных систем и программных технологий

ОТЧЕТ
по лабораторной работе

«Язык SQL-DDL»

Базы данных

Работу выполнил студент

группа 43501/3 Дьячков В.В.

Работу принял преподаватель

_____ Мяснов А.В.

Санкт-Петербург

2018

Содержание

1. Цель работы

Познакомиться с языком создания запросов управления данными SQL-DML.

2. Программа работы

1. Изучение SQL-DML.
2. Выполнение всех запросов из списка стандартных запросов. Демонстрация результатов преподавателю.
3. Получение у преподавателя и реализация SQL-запросов в соответствии с индивидуальным заданием. Демонстрация результатов преподавателю.
4. Сохранение в БД выполненных запросов `SELECT` в виде представлений, запросов `INSERT`, `UPDATE` или `DELETE` – в виде ХП. Выкладывание скрипта в GitLab.

3. Стандартные запросы

3.1. Выборка с использованием логических операций

Задание: Сделайте выборку данных из одной таблицы при нескольких условиях, с использованием логических операций, `LIKE`, `BETWEEN`, `IN` (не менее 3-х разных примеров).

Выборка бронирований указанных после `IN` пользователей:

```
SELECT *
FROM reservation
WHERE user_id IN (1, 2, 3)
  AND "from" < to_date('2017-01-31', 'YYYY-MM-DD');
```

Листинг 1: `select-in.sql`

```
  id  | room_id | user_id |    from    |     to     |   price    | is_paid
------+---------+---------+------------+------------+------------+---------
 1444 |    1526 |       1 | 2017-01-26 | 2017-02-06 |  $5,315.00 | t
 1724 |    4742 |       3 | 2017-01-30 | 2017-02-14 | $12,607.00 | f
 7195 |    2238 |       1 | 2017-01-16 | 2017-01-30 | $10,545.00 | t
(3 rows)
```

Выборка бронирований между датами, указанными после `BETWEEN`:

```
SELECT *
FROM reservation
WHERE price < 100 :: money
  AND "from" BETWEEN to_date('2017-01-01', 'YYYY-MM-DD')
          AND to_date('2017-01-31', 'YYYY-MM-DD');
```

Листинг 2: `select-between.sql`

```
  id   | room_id | user_id |    from    |     to     | price  | is_paid
-------+---------+---------+------------+------------+--------+---------
  4292 |    3256 |     967 | 2017-01-26 | 2017-01-31 | $85.00 | f
  5008 |    1916 |     283 | 2017-01-26 | 2017-02-11 |  $2.00 | t
 13670 |     720 |     914 | 2017-01-14 | 2017-01-17 | $73.00 | t
 16510 |    4480 |     382 | 2017-01-15 | 2017-01-31 | $42.00 | t
(4 rows)
```

Выборка пользователей, имя которых начинается с «вадим» и номер телефона начинается с единицы:

```sql
SELECT id, name, phone_number, city_id
FROM "user"
WHERE name LIKE 'вадим%'
  AND phone_number LIKE '1-%';
```

Листинг 3: `select-like.sql`

```
 id  |      name       |  phone_number   | city_id
-----+-----------------+-----------------+---------
 177 | вадим.фадеев    | 1-075-921-5362  |     110
 320 | вадим.богданова | 1-773-542-8914  |     124
 730 | вадим.попова    | 1-196-683-9202  |     138
(3 rows)
```

3.2. Запрос с вычисляемым полем

Задание: Создайте в запросе вычисляемое поле.

Вычисляемые значения: длительность бронирования и вычисление стоимости бронирования одного дня:

```sql
SELECT id, "to" - "from"                AS duration,
       price / NULLIF("to" - "from", 0) AS per_night
FROM reservation
LIMIT 10;
```

Листинг 4: `select-eval.sql`

```
 id | duration | per_night
----+----------+-----------
  1 |        1 | $1,999.00
  2 |        5 | $1,560.00
  3 |        7 | $1,421.14
  4 |       19 |   $599.57
  5 |       12 |   $488.08
  6 |       17 |   $449.29
  7 |       16 |   $235.06
  8 |        8 |   $749.12
  9 |       19 |   $278.21
 10 |       14 |   $815.28
(10 rows)
```

3.3. Выборка с использованием сортировки

Задание: Сделайте выборку всех данных с сортировкой по нескольким полям.

Выборка бронирований с сортировкой по флагу оплаты заказа в порядке убывания (оплаченные заказы будут выше) и стоимости в порядке убывания.

```
SELECT *
FROM reservation
ORDER BY is_paid DESC, price DESC
LIMIT 10;
```

Листинг 5: `select-order.sql`

```
  id   | room_id | user_id |    from    |     to     |   price    | is_paid
-------+---------+---------+------------+------------+------------+---------
  6663 |     948 |     282 | 2019-10-15 | 2019-10-28 | $14,999.00 | t
 17003 |    4774 |     490 | 2019-12-12 | 2019-12-14 | $14,997.00 | t
  4504 |     438 |     336 | 2017-11-28 | 2017-12-11 | $14,996.00 | t
   455 |    1405 |     837 | 2018-12-08 | 2018-12-08 | $14,996.00 | t
 18251 |     660 |     350 | 2018-11-13 | 2018-12-01 | $14,995.00 | t
  1972 |    2006 |     160 | 2019-07-27 | 2019-08-02 | $14,992.00 | t
 16859 |    3905 |     648 | 2017-06-15 | 2017-06-29 | $14,992.00 | t
  1947 |      71 |      63 | 2018-07-21 | 2018-08-02 | $14,991.00 | t
  7764 |    4147 |     735 | 2017-01-09 | 2017-01-18 | $14,980.00 | t
   929 |    1403 |     681 | 2018-11-28 | 2018-12-14 | $14,979.00 | t
(10 rows)
```

3.4. Запрос с вычислением совокупных характеристик таблиц

Задание: Создайте запрос, вычисляющий несколько совокупных характеристик таблиц.

Выборка с вычисление совокупных характеристик: количества бронирований, средней продолжительности бронирования, средней и максимальной стоимости заказа и суммы стоимости всех бронирований:

```
SELECT COUNT(*)                       AS count,
       AVG("to" - "from")             AS avg_duration,
       AVG(price :: numeric) :: money AS avg_price,
       MAX(price)                     AS max_price,
       SUM(price)                     AS sum_price
FROM reservation;
```

Листинг 6: `select-summary.sql`

```
 count |    avg_duration     | avg_price | max_price  |    sum_price
-------+---------------------+-----------+------------+-----------------
 20000 | 10.0916500000000000 | $7,500.42 | $14,999.00 | $150,008,493.00
(1 row)
```

3.5. Выборка из связанных таблиц

Задание: Сделайте выборку данных из связанных таблиц (не менее двух примеров).

Соединение бронирований, комнат и тип комнат:

```sql
SELECT "from", "to", type, capacity
FROM reservation
     JOIN room ON reservation.room_id = room.id
     JOIN room_type ON room.room_type_id = room_type.id
LIMIT 10;
```

Листинг 7: `select-join-1.sql`

```
    from    |     to     |   type    | capacity
------------+------------+-----------+----------
 2019-06-17 | 2019-06-18 | Hestia    |        1
 2018-08-30 | 2018-09-04 | Zeus      |        1
 2018-07-17 | 2018-07-24 | Ares      |        4
 2018-01-18 | 2018-02-06 | Artemis   |        3
 2018-03-03 | 2018-03-15 | Aphrodite |        1
 2019-10-03 | 2019-10-20 | Hermes    |        2
 2019-07-29 | 2019-08-14 | Hestia    |        5
 2017-08-18 | 2017-08-26 | Dionysus  |        5
 2017-07-19 | 2017-08-07 | Hades     |        5
 2018-08-05 | 2018-08-19 | Hestia    |        5
(10 rows)
```

Соединение пользователей, городов и стран:

```sql
SELECT usr.name, city.name AS city, country.name AS country
FROM "user" AS usr
     JOIN city ON usr.city_id = city.id
     JOIN country ON city.country_id = country.id
LIMIT 10;
```

Листинг 8: `select-join-2.sql`

```
       name        |    city     |              country
-------------------+-------------+---------------------------------
 светлана.авдеева  | Владивосток | Никарагуа
 валентин.панфилов | Красноярск  | Габон
 вадим.захарова    | Рязань      | Великобритания
 владимир.беспалов | Иваново     | Великобритания
 анжела.некрасова  | Воронеж     | Фарерские Острова (не признана)
 дмитрий.туров     | Омск        | Великобритания
 андрей.щербакова  | Сочи        | Кирибати
 фёдор.наумова     | Иваново     | Лесото
 никита.щербаков   | Брянск      | Узбекистан
 ангелина.кошелев  | Кемерово    | Лесото
(10 rows)
```

3.6. Запрос с использованием группировки

Задание: Создайте запрос, рассчитывающий совокупную характеристику с использованием группировки, наложите ограничение на результат группировки.

Группировка бронирований по пользователю с накладываемым условием, что число бронирований у пользователя больше 30:

```sql
SELECT user_id
FROM reservation
GROUP BY user_id
HAVING COUNT(1) > 30;
```

Листинг 9: `select-group.sql`

```
 user_id
---------
     663
     745
     593
       8
     846
     133
     337
     163
     851
(9 rows)
```

3.7. Вложенный запрос

Задание: Придумайте пример использования вложенного запроса.

Вложенный запрос: выборка пользователей, имеющих более 30 бронирований, и соединение с таблицей пользователей после этого:

```sql
SELECT usr.id, usr.name, q.cnt
FROM (SELECT user_id, COUNT(1) AS cnt
      FROM reservation
      GROUP BY user_id
      HAVING COUNT(1) > 30
     ) q
     JOIN "user" AS usr ON q.user_id = usr.id;
```

Листинг 10: `select-subquery.sql`

```
 id  |        name         | cnt
-----+---------------------+-----
   8 | фёдор.наумова       |  31
 133 | евгения.елисеева    |  31
 163 | даниил.субботина    |  32
 337 | клавдия.трофимова   |  33
 593 | ирина.лазарев       |  33
 663 | георгий.степанова   |  37
 745 | юрий.анисимов       |  32
 846 | ольга.титова        |  31
 851 | георгий.виноградова |  31
(9 rows)
```

3.8. Вставка записей

Задание: С помощью оператора `INSERT` добавьте в каждую таблицу по одной записи.

```
INSERT INTO house_rules
VALUES (DEFAULT, '15:00:00', '12:00:00', 'Предоплата не возвращается при отмене бронирования менее чем за сутки'),
       (DEFAULT, '16:00:00', '14:00:00', 'Предоплата возвращается');

INSERT INTO hotel
VALUES (DEFAULT, 'Гранд Будапешт', 1, 'Республика Зубровка', 5, 'Комфортабельный отель Гранд Будапешт');

INSERT INTO "user"
VALUES (DEFAULT, 'traveller', 'traveller@travel.com', '5E884898DA28047151D0E56F8DC6292773603D0D6AABBDD62A11EF721D1542D8');

INSERT INTO room_type
VALUES (DEFAULT, 1, 'double king-size', 2, 'Номер для некурящих с 2 кроватями размера king-size'),
       (DEFAULT, 1, 'single', 1, 'Номер для некурящих с 1 кроватью');

INSERT INTO room
VALUES (DEFAULT, 1, '№123'),
       (DEFAULT, 2, '№456'),
       (DEFAULT, 1, '№789');

INSERT INTO reservation
VALUES (DEFAULT, 1, 1, '2018-10-08', '2018-10-10', 1234, TRUE),
       (DEFAULT, 2, 1, '2018-10-10', '2018-10-15', 4321, FALSE);

INSERT INTO guest
VALUES (DEFAULT, 1, 'M. Gustave', FALSE),
       (DEFAULT, 1, 'Mr. Moustafa', FALSE),
       (DEFAULT, 2, 'Serge X.', FALSE);

INSERT INTO review
VALUES (DEFAULT, 1, 'Все хорошо', 'Все плохо', 3),
       (DEFAULT, 2, 'Все плохо', 'Все хорошо', 5);

INSERT INTO price
VALUES (DEFAULT, 1, '2018-10-01', '2018-10-31', 1234),
       (DEFAULT, 1, '2018-09-01', '2018-09-30', 1000),
       (DEFAULT, 2, '2018-10-01', '2018-10-31', 4321);

INSERT INTO facility
VALUES (DEFAULT, 'Wi-Fi'),
       (DEFAULT, 'Холодильник'),
       (DEFAULT, 'TV'),
       (DEFAULT, 'Душ');

INSERT INTO room_facility
VALUES (1, 2),
       (1, 3),
       (1, 4),
       (2, 1),
       (2, 4);
```

Листинг 11: `insert.sql`

3.9. Изменение записей

Задание: С помощью оператора `UPDATE` измените значения нескольких полей у всех записей, отвечающих заданному условию.

Выставление цены бронирования, равной 1$, в строках, где цена оказалась равна 0$. Для наглядности до и после добавим запросы выборки таких строк:

```
SELECT *
FROM reservation
WHERE price = 0 :: money;

UPDATE reservation
SET price = 0 :: money
WHERE price = 1 :: money;

SELECT *
FROM reservation
WHERE price = 0 :: money;
```

Листинг 12: `update.sql`

```
  id   | room_id | user_id |    from    |     to     | price | is_paid
-------+---------+---------+------------+------------+-------+---------
  9005 |    4836 |     163 | 2019-04-09 | 2019-04-12 | $0.00 | t
  2679 |    2357 |     576 | 2017-11-15 | 2017-11-17 | $0.00 | f
 16017 |    2548 |     566 | 2017-03-07 | 2017-03-11 | $0.00 | t
 16811 |    2256 |     547 | 2019-12-30 | 2020-01-16 | $0.00 | f
(4 rows)

UPDATE 4
 id | room_id | user_id | from | to | price | is_paid
----+---------+---------+------+----+-------+---------
(0 rows)
```

3.10. Удаление записей по условию

Задание: С помощью оператора `DELETE` удалите запись, имеющую максимальное (минимальное) значение некоторой совокупной характеристики.

Удаление строки из таблицы цен, имеющую максимальную цену. Для наглядности до и после добавим запросы выборки таких строк:

```
SELECT *
FROM price
WHERE price.price = (SELECT MAX(price.price) FROM price);

DELETE
FROM price
WHERE price.price = (SELECT MAX(price.price) FROM price);

SELECT *
FROM price
WHERE price.price = (SELECT MAX(price.price) FROM price);
```

Листинг 13: `delete-1.sql`

```
  id  | room_type_id |    from    |     to     |   price
------+--------------+------------+------------+------------
 4904 |          207 | 2018-10-16 | 2018-10-17 | $14,992.00
(1 row)

DELETE 1
  id  | room_type_id |    from    |     to     |   price
------+--------------+------------+------------+------------
 9493 |          486 | 2017-09-05 | 2017-09-21 | $14,987.00
(1 row)
```

3.11. Удаление с использованием вложенного запроса

Задание: С помощью оператора DELETE удалите записи в главной таблице, на которые не ссылается подчиненная таблица (используя вложенный запрос).

Удаление пользователей, не имеющих бронирований. Для наглядности до и после добавим запросы выборки таких строк:

```sql
SELECT usr.id, usr.name
FROM "user" usr
WHERE usr.id NOT IN (SELECT user_id
                     FROM reservation
                     GROUP BY user_id
                    );

DELETE
FROM "user" AS usr
WHERE usr.id NOT IN (SELECT user_id
                     FROM reservation
                     GROUP BY user_id
                    );

SELECT usr.id, usr.name
FROM "user" usr
WHERE usr.id NOT IN (SELECT user_id
                     FROM reservation
                     GROUP BY user_id
                    );
```

Листинг 14: `delete-2.sql`

```
  id  | name
------+------
 1003 | test
 1004 | test
(2 rows)

DELETE 2
 id | name
----+------
(0 rows)
```

4. Запросы в соответствие с задание преподавателя

4.1. Рейтинг городов по кварталам

Задание: Вывести рейтинг городов по кварталам. В рейтинге 5 городов в, которые больше всего ездят в каком-то квартале.

Будем формировать итоговый запрос поэтапно.

1. Формирование отчета посещаемости городов по кварталам. Для удобства упорядочим по городу и кварталу и возьмем записи для первых двух городов:

```sql
SELECT city.id                       AS city_id,
       EXTRACT(QUARTER FROM "from") AS quarter,
       COUNT(*)                      AS count
FROM reservation
  JOIN room ON room.id = reservation.room_id
  JOIN room_type ON room_type.id = room.room_type_id
  JOIN hotel ON hotel.id = room_type.hotel_id
  JOIN city ON hotel.city_id = city.id
GROUP BY city.id, quarter
ORDER BY city.id, quarter
LIMIT 8
```

Листинг 15: `quarter-quarter-summary.sql`

```
 city_id | quarter | count
---------+---------+-------
       1 |       1 |     9
       1 |       2 |    13
       1 |       3 |    13
       1 |       4 |    10
       2 |       1 |    33
       2 |       2 |    36
       2 |       3 |    25
       2 |       4 |    36
(8 rows)
```

2. Формирование порядкового номера полученной строки посещаемости города относительно других городов в рамках квартала. Для этого используется оконная функция `ROW_NUMBER()`, условие `PARTITION BY` разбивает строки на наборы меньшего набора (в нашем случае разбивает по кварталам), а `ORDER BY` указывает на порядок внутри этого набора (в нашем случае по уменьшению бронирований). Для удобства так же возьмем записи о первых двух городах:

```sql
WITH quarter_summary AS (
    SELECT city.id                       AS city_id,
           EXTRACT(QUARTER FROM "from") AS quarter,
           COUNT(*)                      AS count
    FROM reservation
        JOIN room ON room.id = reservation.room_id
        JOIN room_type ON room_type.id = room.room_type_id
        JOIN hotel ON hotel.id = room_type.hotel_id
        JOIN city ON hotel.city_id = city.id
    GROUP BY city.id, quarter
```

```
)
SELECT ROW_NUMBER() OVER (PARTITION BY quarter ORDER BY SUM(count) DESC) AS row_num,
       city_id                                                           AS city_id,
       quarter                                                           AS quarter,
       SUM(count)                                                        AS reservations
FROM quarter_summary
GROUP BY quarter, city_id
ORDER BY city_id, quarter
LIMIT 8
```

Листинг 16: `quarter-summary.sql`

```
 row_num | city_id | quarter | reservations
---------+---------+---------+--------------
     148 |       1 |       1 |            9
     130 |       1 |       2 |           13
     134 |       1 |       3 |           13
     141 |       1 |       4 |           10
      57 |       2 |       1 |           33
      58 |       2 |       2 |           36
      89 |       2 |       3 |           25
      53 |       2 |       4 |           36
(8 rows)
```

3. Формирование итогового рейтинга городов по кварталам. Для этого из предыдущего запроса берутся строки с порядковым номером 1, что соответствует ID городов, в которых забронировано больше всего номеров в одном из кварталов. После этого по ID соединяются города и страны:

```
WITH summary AS (
    WITH quarter_summary AS (
        SELECT city.id                       AS city_id,
               EXTRACT(QUARTER FROM "from") AS quarter,
               COUNT(*)                      AS count
        FROM reservation
            JOIN room ON room.id = reservation.room_id
            JOIN room_type ON room_type.id = room.room_type_id
            JOIN hotel ON hotel.id = room_type.hotel_id
            JOIN city ON hotel.city_id = city.id
        GROUP BY city.id, quarter
    )
    SELECT ROW_NUMBER() OVER (PARTITION BY quarter ORDER BY SUM(count) DESC) AS row_num,
           city_id                                                           AS city_id,
           quarter                                                           AS quarter,
           SUM(count)                                                        AS reservations
    FROM quarter_summary
    GROUP BY city_id, quarter
)
SELECT summary.quarter      AS quarter,
       city.name            AS city_name,
       country.name         AS country_name,
       summary.reservations AS reservations
FROM summary
         JOIN city ON summary.city_id = city.id
         JOIN country ON city.country_id = country.id
WHERE summary.row_num < 2
ORDER BY summary.quarter, summary.reservations DESC;
```

Листинг 17: `quarter.sql`

```
 quarter |   city_name     | country_name | reservations
---------+-----------------+--------------+--------------
       1 | Ростов-на-Дону | Узбекистан   |          115
       2 | Ростов-на-Дону | Узбекистан   |          122
       3 | Ростов-на-Дону | Узбекистан   |          109
       4 | Воронеж         | Парагвай     |          107
(4 rows)
```

4.2. Клиенты, имеющие наибольший средний рост стоимости путевки

Задание: Вывести 5 клиентов, которые имеют наибольший средний рост стоимости путевки.

Будем формировать итоговый запрос поэтапно.

1. Формирование порядкового номера бронирования относительно бронирований отдельного пользователя. Для этого так же используется оконная функция `ROW_NUMBER()` с ID пользователя как условием разбивания на наборы и датой начала бронирования для упорядочивания. Для анализа используются только пользователи, имеющие более 10 заказов. Для удобства возьмем только первые 10 строк:

```sql
SELECT ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY "from") AS row_num,
       price                                                    AS price,
       user_id                                                  AS user_id
FROM reservation
WHERE is_paid AND user_id IN (
    SELECT user_id
    FROM reservation
    WHERE is_paid
    GROUP BY user_id
    HAVING COUNT(*) > 10
)
LIMIT 10
```

Листинг 18: `travellers-increase-summary.sql`

```
 row_num |   price    | user_id
---------+------------+---------
       1 | $10,545.00 |       1
       2 |  $5,315.00 |       1
       3 | $12,319.00 |       1
       4 |  $8,461.00 |       1
       5 | $14,906.00 |       1
       6 | $11,208.00 |       1
       7 |  $7,457.00 |       1
       8 | $14,935.00 |       1
       9 | $11,365.00 |       1
      10 |  $5,193.00 |       1
(10 rows)
```

2. Формирование для каждого пользователя средней разницы между стоимостью соседних (относительно даты) бронирований и выбор 5 клиентов с наибольшим этим показателем:

```sql
WITH increase_summary AS (
    SELECT ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY "from") AS row_num,
           price                                                    AS price,
           user_id                                                  AS user_id
    FROM reservation
    WHERE is_paid AND user_id IN (
        SELECT user_id
        FROM reservation
        WHERE is_paid
        GROUP BY user_id
        HAVING COUNT(*) > 10
    )
)
SELECT curr.user_id                                          AS user_id,
       AVG((curr.price - prev.price) :: numeric) :: money AS avg_diff
FROM increase_summary curr
     JOIN increase_summary prev
     ON curr.user_id = prev.user_id AND curr.row_num = prev.row_num + 1
GROUP BY curr.user_id
ORDER BY avg_diff DESC
LIMIT 5
```

Листинг 19: `travellers-summary.sql`

```
 user_id | avg_diff
---------+-----------
     183 | $1,294.90
     578 | $1,279.00
     536 | $1,207.30
     281 | $1,199.18
     778 | $1,165.42
(5 rows)
```

3. Формирование сводной таблицы 5 клиентов, имеющих наибольший средний рост стоимости путевки: ID и имя пользователя, количество бронирований, суммарная стоимость бронирований и средняя разница между бронированиями:

```sql
WITH summary AS (
    WITH increase_summary AS (
        SELECT ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY "from") AS row_num,
               price                                                    AS price,
               user_id                                                  AS user_id
        FROM reservation
        WHERE is_paid AND user_id IN (
            SELECT user_id
            FROM reservation
            WHERE is_paid
            GROUP BY user_id
            HAVING COUNT(*) > 10
        )
    )
    SELECT curr.user_id                                          AS user_id,
           AVG((curr.price - prev.price) :: numeric) :: money AS avg_diff
    FROM increase_summary curr
         JOIN increase_summary prev
         ON curr.user_id = prev.user_id AND curr.row_num = prev.row_num + 1
    GROUP BY curr.user_id
```

```
    ORDER BY avg_diff DESC
    LIMIT 5
)
SELECT usr.id, usr.name, reservations, total, avg_diff
FROM summary
     JOIN "user" AS usr ON user_id = usr.id
     JOIN (
         SELECT user_id    AS user_id,
                COUNT(*)   AS reservations,
                SUM(price) AS total
         FROM reservation
         GROUP BY user_id
     ) AS total ON total.user_id = usr.id;
```

Листинг 20: `travellers.sql`

```
 id  |       name         | reservations |    total     | avg_diff
-----+--------------------+--------------+--------------+----------
 578 | лидия.соловьева    |           25 | $177,712.00 | $1,279.00
 778 | варвара.корнилова  |           20 | $171,200.00 | $1,165.42
 281 | анфиса.крылов      |           25 | $157,819.00 | $1,199.18
 183 | лидия.аксенов      |           24 | $184,655.00 | $1,294.90
 536 | дмитрий.воробьев   |           26 | $201,855.00 | $1,207.30
(5 rows)
```

5. Представления и хранимые процедуры

Оформим стандартные запросы и запросы, заданные преподавателем, в виде представлений. Представления – виртуальные именованные таблицы, создаваемые с помощью запроса SELECT.

```
CREATE VIEW select_in AS
    SELECT *
    FROM reservation
    WHERE user_id IN (1, 2, 3)
      AND "from" < to_date('2017-01-31', 'YYYY-MM-DD');

CREATE VIEW select_between AS
    SELECT *
    FROM reservation
    WHERE price < 100 :: money
      AND "from" BETWEEN to_date('2017-01-01', 'YYYY-MM-DD')
              AND to_date('2017-01-31', 'YYYY-MM-DD');

CREATE VIEW select_like AS
    SELECT id, name, phone_number, city_id
    FROM "user"
    WHERE name LIKE 'вадим%'
      AND phone_number LIKE '1-%';

CREATE VIEW select_eval AS
    SELECT id, "to" - "from"                AS duration,
           price / NULLIF("to" - "from", 0) AS per_night
    FROM reservation
    LIMIT 10;

CREATE VIEW select_order AS
```

```sql
    SELECT *
    FROM reservation
    ORDER BY is_paid DESC, price DESC
    LIMIT 10;

CREATE VIEW select_summary AS
    SELECT COUNT(*)                       AS count,
           AVG("to" - "from")             AS avg_duration,
           AVG(price :: numeric) :: money AS avg_price,
           MAX(price)                     AS max_price,
           SUM(price)                     AS sum_price
    FROM reservation;

CREATE VIEW select_join_1 AS
    SELECT "from", "to", type, capacity
    FROM reservation
         JOIN room ON reservation.room_id = room.id
         JOIN room_type ON room.room_type_id = room_type.id
    LIMIT 10;

CREATE VIEW select_join_2 AS
    SELECT usr.name, city.name AS city, country.name AS country
    FROM "user" AS usr
         JOIN city ON usr.city_id = city.id
         JOIN country ON city.country_id = country.id
    LIMIT 10;

CREATE VIEW select_group AS
    SELECT user_id
    FROM reservation
    GROUP BY user_id
    HAVING COUNT(1) > 30;

CREATE VIEW select_subquery AS
    SELECT usr.id, usr.name, q.cnt
    FROM (SELECT user_id, COUNT(1) AS cnt
          FROM reservation
          GROUP BY user_id
          HAVING COUNT(1) > 30
         ) q
         JOIN "user" AS usr ON q.user_id = usr.id;

CREATE VIEW "quarter" AS
    WITH summary AS (
        WITH quarter_summary AS (
            SELECT city.id                       AS city_id,
                   EXTRACT(QUARTER FROM "from") AS quarter,
                   COUNT(*)                      AS count
            FROM reservation
                JOIN room ON room.id = reservation.room_id
                JOIN room_type ON room_type.id = room.room_type_id
                JOIN hotel ON hotel.id = room_type.hotel_id
                JOIN city ON hotel.city_id = city.id
            GROUP BY city.id, quarter
        )
        SELECT ROW_NUMBER() OVER (PARTITION BY quarter ORDER BY SUM(count) DESC) AS row_num,
               city_id                                                          AS city_id,
               quarter                                                          AS quarter,
               SUM(count)                                                       AS
    reservations
```

```
86          FROM quarter_summary
87          GROUP BY city_id, quarter
88      )
89      SELECT summary.quarter      AS quarter,
90             city.name            AS city_name,
91             country.name         AS country_name,
92             summary.reservations AS reservations
93      FROM summary
94               JOIN city ON summary.city_id = city.id
95               JOIN country ON city.country_id = country.id
96      WHERE summary.row_num < 2
97      ORDER BY summary.quarter, summary.reservations DESC;
98
99  CREATE VIEW travelling AS
100     WITH summary AS (
101         WITH increase_summary AS (
102             SELECT ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY "from") AS row_num,
103                    price                                                   AS price,
104                    user_id                                                 AS user_id
105             FROM reservation
106             WHERE is_paid AND user_id IN (
107                 SELECT user_id
108                 FROM reservation
109                 WHERE is_paid
110                 GROUP BY user_id
111                 HAVING COUNT(*) > 10
112             )
113         )
114         SELECT curr.user_id                                          AS user_id,
115                AVG((curr.price - prev.price) :: numeric) :: money AS avg_diff
116         FROM increase_summary curr
117              JOIN increase_summary prev
118              ON curr.user_id = prev.user_id AND curr.row_num = prev.row_num + 1
119         GROUP BY curr.user_id
120         ORDER BY avg_diff DESC
121         LIMIT 5
122     )
123     SELECT usr.id, usr.name, reservations, total, avg_diff
124     FROM summary
125          JOIN "user" AS usr ON user_id = usr.id
126          JOIN (
127              SELECT user_id    AS user_id,
128                     COUNT(*)   AS reservations,
129                     SUM(price) AS total
130              FROM reservation
131              GROUP BY user_id
132          ) AS total ON total.user_id = usr.id;
```

Листинг 21: `views.sql`

Оформим запросы изменения и удаления данных как хранимые процедуры. Однако, хранимые процедуры появятся только в PostgreSQL11, поэтому оформим не как процедуры, а как функции, возвращающие void.

```
1  CREATE FUNCTION IncreaseZeroPrice()
2      RETURNS void AS
3  $$
4  BEGIN
5      UPDATE reservation
6      SET price = 1 :: money
```

```
7       WHERE price = 0 :: money;
8   END;
9   $$
10  LANGUAGE plpgsql;
11
12  CREATE FUNCTION DeleteMaxPrice()
13      RETURNS void AS
14  $$
15  BEGIN
16      DELETE
17      FROM price
18      WHERE price.price = (SELECT MAX(price.price)
19                           FROM price
20                          );
21  END;
22  $$
23  LANGUAGE plpgsql;
24
25  CREATE FUNCTION DeleteUsersWithoutReservations()
26      RETURNS void AS
27  $$
28  BEGIN
29      DELETE
30      FROM "user" AS usr
31      WHERE usr.id NOT IN (SELECT user_id
32                           FROM reservation
33                           GROUP BY user_id
34                          );
35  END;
36  $$
37  LANGUAGE plpgsql;
```

Листинг 22: `functions.sql`

6. Выводы

В процессе выполнения данной работы:

- изучен SQL-DML;
- выполнены стандартные запросы выборки данных: с условиями, вычисляемыми полями, совокупными характеристиками, с соединениями, с использованием группировки и сортировки;
- выполнены запросы вставки тестовых данных;
- выполнены запросы удаления данных по различным условиям;
- выполнены запросы выборки данных, заданные преподавателем: формирование рейтинга городов по кварталам и выбор 5 клиентов, имеющих наибольший средний рост стоимости путевки;
- запросы выборки оформлены как представления, а изменения и удаления как хранимые процедуры.