**Представления для просмотра данных:**

1. Список корпусов гостиницы.
2. Перечень номеров с указанием корпуса и характеристик.
3. Бронь на будущее время.
4. Список истории проживания.
5. Список дополнительных услуг с указанием цены.
6. Список клиентов с общим количеством дней проживания, общей стоимостью проживания, датой окончания последнего проживания.
7. Сделать отбор Топ 5 клиентов / Топ 5 партнеров с количеством проживаний и общим количеством дней проживания.
8. Список партнеров с общим количеством дней проживания, общей стоимостью проживания, датой окончания последнего проживания (для всех сотрудников партнера).
9. Список клиентов-должников.
10. Перечень жалоб.

**Представления со смысловой нагрузкой:**

1. Список номеров с количеством проживаний и общим количеством дней проживания
2. Сделать отборы по корпусу, ТОП 3 лучших номеров (наиболее посещаемых).
3. Список клиентов, ни разу не воспользовавшимися дополнительными услугами.
4. Список клиентов, пользовавшихся дополнительными услугами.
5. Список клиентов, ни разу не оставлявших жалобы.
6. Список клиентов, оставивших хотя бы одну жалобу.

**Функции:**

1. Получить перечень и общее число клиентов, заселявшихся в номера с указанными характеристиками за некоторый период.
2. Получить список жалоб на проживание в определенном корпусе и от определенного клиента.
3. Получить список и количество свободных номеров и количество свободных мест на данный момент, на определенную дату.
4. Получить сведения о количестве свободных номеров: по корпусам; по количеству мест в номере.
5. Вывести сведения о занятости и свободном времени для указанного номера. (пока без свободного времени)
6. Получить список занятых сейчас номеров, которые освобождаются к указанному сроку.
7. Получить данные об объеме бронирования номеров конкретным партнером за указанный период: в целом и с разбивкой и номерам.
8. Получить сведения о партнерах, с которыми заключены договора о брони на указанный период.
9. Получить сведения о новых клиентах за указанный период.
10. Получить сведения о конкретном клиенте гостиницы: в каких номерах и в какой период останавливался.
11. Получить сведения о конкретном номере: кем он был занят в определенный период.
12. Получить процентное отношение всех номеров к номерам, бронируемым партнерами, за определенный период.
13. Вывести график работы конкретного сотрудника.
14. Вывести все брони, оформленные указанным сотрудником.
15. С разбивкой по сотрудникам получить количество отработанных дней и часов за определенный период.

**Процедуры:**

1. Установить график работы сотруднику. (alter table set)
2. Увеличить цены всех услуг на заданное количество процентов.

**Триггеры:**

1. Автоматически изменять статус номера при создании / отмене брони.
2. Автоматически изменять поле “Задолженность” при внесении платежа.

**Виды запросов в информационной системе:**

1. Получить перечень и общее число фирм, забронировавших места в объеме, не менее указанного, за весь период сотрудничества, либо за некоторый период.
2. Получить перечень и общее число постояльцев, заселявшихся в номера с указанными характеристиками за некоторый период.
3. Получить количество свободных номеров на данный момент.
4. Получить сведения о количестве свободных номеров с указанными характеристиками.
5. Получить сведения о конкретном свободном номере: в течение какого времени он будет пустовать и о его характеристиках.
6. Получить список занятых сейчас номеров, которые освобождаются к указанному сроку.
7. Получить данные об объеме бронирования номеров данной фирмой за указанный период, и каким номерам отдавались предпочтения.
8. Получить список недовольных клиентов и их жалобы.
9. Получить данные о рентабельности номеров с определенными характеристиками: соотношение об объеме продаж номеров к накладным расходам за указанный период.
10. Получить сведения о постояльце из заданного номера: его счет гостинице за дополнительные услуги, поступавшие от него жалобы, виды дополнительных услуг, которыми он пользовался.
11. Получить сведения о фирмах, с которыми заключены договора о брони на указанный период.
12. Получить сведения о наиболее часто посещающих гостиницу постояльцах по [всем корпусам гостиниц](http://topuch.com/lekcii-1-klassifikaciya-gostinic-za-rubejom/index.html), по определенному зданию.
13. Получить сведения о новых клиентах за указанный период.
14. Получить сведения о конкретном человеке, сколько раз он посещал гостиницу, в каких номерах и в какой период останавливался, какие счета оплачивал.
15. Получить сведения о конкретном номере: кем он был занят в определенный период.
16. Получить процентное отношение всех номеров к номерам, бронируемым партнерами.

|  |  |
| --- | --- |
| **Представления для просмотра данных:** | |
| **1** | select \* from hotel |
| **2** | select \* from rooms |
| **3** | select \* from reservations  where in\_data > now() |
| **4** | select \* from history |
| **5** | select \* from services |
| **6** | SELECT c.id, surname, name, patronymic, SUM(check\_out\_data - (check\_in\_data-1)) as days, SUM(payment), MAX(check\_out\_data) FROM clients c  LEFT JOIN history h ON c.id = h.client\_id  GROUP BY c.id, surname, name, patronymic, client\_id  order by id |
| **7** | CREATE TEMPORARY TABLE temp\_table AS (SELECT CONCAT(surname, ' ', name, ' ', patronymic) AS "Top 5 partners/clients", COUNT(check\_in\_data) AS check\_ins, SUM(check\_out\_data - (check\_in\_data-1)) as days FROM clients c LEFT JOIN history h ON c.id=h.client\_id GROUP BY c.id, client\_id HAVING SUM(check\_out\_data - (check\_in\_data-1)) > 0 ORDER BY days DESC LIMIT 5)  UNION  (SELECT organization::text, COUNT(check\_in\_data) AS check\_ins, SUM(check\_out\_data - (check\_in\_data-1)) as days FROM partners p LEFT JOIN history h ON p.id=h.partner\_id GROUP BY organization HAVING SUM(check\_out\_data - (check\_in\_data-1)) > 0 ORDER BY days DESC LIMIT 5);  SELECT \* FROM temp\_table; |
| **8** | SELECT organization::text, SUM(check\_out\_data - (check\_in\_data-1)) as days, Sum(payment), (select check\_out\_data) as "last day" FROM partners p  LEFT JOIN history h ON p.id=h.partner\_id  GROUP BY organization,check\_out\_data  HAVING SUM(check\_out\_data - (check\_in\_data-1)) > 0  ORDER BY days |
| **9** | select \* from clients  where id in (select client\_id from history where debt - payment > 0) |
| **10** | select \* from complaints\_list |

|  |  |
| --- | --- |
| **Представления со смысловой нагрузкой:** | |
| **1** | SELECT room\_number, COUNT(check\_in\_data) AS number\_of\_stays, SUM(check\_out\_data - (check\_in\_data - 1)) AS total\_stay\_days FROM history  GROUP BY room\_number  order by room\_number |
| **2** | CREATE OR REPLACE FUNCTION top\_rooms()  RETURNS TABLE (corpus TEXT, room INT, days BIGINT)  AS $$  BEGIN  RETURN QUERY  SELECT corp, room\_number, dayss  FROM (  SELECT LEFT(room\_number::text, 1) AS corp, h.room\_number, SUM(h.check\_out\_data - (h.check\_in\_data-1)) as dayss,  ROW\_NUMBER() OVER (PARTITION BY LEFT(room\_number::text, 1) ORDER BY SUM(h.check\_out\_data - (h.check\_in\_data-1)) DESC) as rn  FROM history h  GROUP BY LEFT(room\_number::text, 1), h.room\_number  ) sub  WHERE rn <= 3  ORDER BY corpus, days desc;  END;  $$ LANGUAGE plpgsql;  select \* from top\_rooms() |
| **3** | select CONCAT(surname, ' ', name, ' ', patronymic) AS clients from clients  where id in (select client\_id from history  where id not in (select history\_id from services\_list group by history\_id)) |
| **4** | select CONCAT(surname, ' ', name, ' ', patronymic) AS clients from clients  where id in (select client\_id from history  where id in (select history\_id from services\_list group by history\_id)) |
| **5** | select CONCAT(surname, ' ', name, ' ', patronymic) AS clients from clients  where id in (select client\_id from history  where id not in (select history\_id from complaints\_list group by history\_id)) |
| **6** | select CONCAT(surname, ' ', name, ' ', patronymic) AS clients from clients  where id in (select client\_id from history  where id not in (select history\_id from complaints\_list group by history\_id)) |

|  |  |
| --- | --- |
| **Функции:** | |
| **1** | CREATE OR REPLACE FUNCTION Rooms\_info(start\_date DATE, end\_date DATE, room\_size INT, room\_build INT)  RETURNS varchar (100)  AS $$  DECLARE  curs1 cursor for SELECT CONCAT(surname, ' ', name, ' ', patronymic) FROM history h  JOIN rooms r ON h.room\_number = r.id  JOIN clients c ON c.id = h.client\_id  WHERE h.check\_in\_data BETWEEN '2024-07-01' AND '2024-07-08' and  r.capacity = 1 and  r.building = 3  group by h.client\_id, surname, name, patronymic;  quantity INT := 0;  cl varchar(50);  BEGIN  open curs1;  fetch curs1 into cl;  loop  EXIT WHEN NOT FOUND;  RAISE NOTICE '%', cl;  quantity:= quantity + 1;  fetch curs1 into cl;  end loop;  RAISE NOTICE 'Количество клиентов заселявшихся в номера с указанными характеристиками: %', quantity;  close curs1;  END;  $$ LANGUAGE plpgsql;  select \* from Rooms\_info('2024-07-01','2024-07-08',1,3) |
| **2** | CREATE OR REPLACE FUNCTION get\_complaints\_list(period\_start DATE, period\_end DATE, building\_id INT, client INT)  RETURNS TABLE ( history\_id INT, data DATE, description TEXT, status varchar(50) )  AS $$  BEGIN  RETURN QUERY SELECT cl.history\_id, cl.data, cl.description, cl.status  FROM complaints\_list cl  JOIN history h ON h.id = cl.history\_id  WHERE (cl.data BETWEEN period\_start AND period\_end)  AND ((SELECT CAST(room\_number AS CHAR)::INT FROM history where id = cl.history\_id) = building\_id)  AND (h.client\_id = client);  END;  $$ LANGUAGE 'plpgsql';  select \* from get\_complaints\_list('2024-07-01', '2024-07-31', 3, 2) |
| **3** | CREATE OR REPLACE FUNCTION available\_rooms(check\_date DATE)  RETURNS TABLE (room INT, capacity INT)  AS $$  BEGIN  IF check\_date = CURRENT\_DATE THEN  RETURN QUERY  SELECT r.id, r.capacity  FROM rooms r  WHERE r.status = 'свободен'  UNION  SELECT  COUNT(CASE WHEN r.status = 'свободен' THEN 1 END)::INT,  SUM(CASE WHEN r.status = 'свободен' THEN r.capacity ELSE 0 END)::INT  FROM rooms r  WHERE r.status = 'свободен'  ORDER BY 1,2 DESC;  ELSE  RETURN QUERY  SELECT r.id, r.capacity  FROM rooms r  WHERE r.booking\_id IS NULL OR NOT EXISTS (SELECT res.id FROM reservations res  WHERE res.in\_data <= check\_date AND res.out\_data > check\_date AND res.id = r.booking\_id)  UNION  SELECT COUNT(r.id)::INT, SUM(r.capacity)::INT  FROM rooms r  WHERE r.booking\_id IS NULL OR NOT EXISTS (SELECT res.id FROM reservations res  WHERE res.in\_data <= check\_date AND res.out\_data > check\_date AND res.id = r.booking\_id)  ORDER BY 1,2 DESC;  END IF;  END;  $$ LANGUAGE plpgsql;  SELECT \* FROM available\_rooms('2024-07-16'); |
| **4** | CREATE OR REPLACE FUNCTION free\_rooms()  RETURNS TABLE (building INT, capacity INT, num\_free\_rooms BIGINT) AS  $$  BEGIN  RETURN QUERY  SELECT r.building, r.capacity, COUNT(r.capacity) AS num\_free\_rooms  FROM rooms r  WHERE status = 'свободен'  GROUP BY r.building, r.capacity;  END;  $$  LANGUAGE 'plpgsql';  select \* from free\_rooms() |
| **5** | **CREATE FUNCTION GetAvailabilityInfo(roomNumber INT)**  **RETURNS TABLE**  **AS $$**  **SELECT**  **check\_in\_data AS occupied\_from,**  **check\_out\_data AS occupied\_until**  **FROM**  **history**  **WHERE**  **room\_number = roomNumber**  **ORDER BY**  **check\_in\_data ASC;**  **$$**  **LANGUAGE SQL;**  **SELECT \* FROM GetAvailabilityInfo(123);** |
| **6** | CREATE OR REPLACE FUNCTION rooms\_get\_free (data DATE)  RETURNS TABLE (  room\_number INT  )  AS $$  BEGIN  RETURN QUERY  SELECT id FROM rooms  WHERE out\_data <= data;  END;  $$ LANGUAGE plpgsql;  select \* from rooms\_get\_free('2024-07-11') |
| **7** | CREATE OR REPLACE FUNCTION partner\_booking (  org VARCHAR,  start\_date DATE,  end\_date DATE)  RETURNS TABLE (  room INT,  booking\_count INT)  AS $$  BEGIN  RETURN QUERY  SELECT  room\_number,  COUNT(\*)::INT  FROM reservations r  JOIN partners p ON r.partner\_id = p.id  WHERE p.organization = org  AND in\_data >= start\_date  AND out\_data <= end\_date  GROUP BY room\_number  UNION  SELECT  NULL,  COUNT(\*)::INT  FROM (  SELECT room\_number  FROM reservations r  JOIN partners p ON r.partner\_id = p.id  WHERE p.organization = 'Турагентство "Эксклюзив"'  AND in\_data >= '2024-07-01'  AND out\_data <= '2024-07-31'  GROUP BY room\_number);  END;  $$ LANGUAGE plpgsql;  select \* from partner\_booking('Турагентство "Эксклюзив"','2024-07-01','2024-07-31') |
| **8** | CREATE OR REPLACE FUNCTION partner\_booking(start\_date DATE, end\_date DATE)  RETURNS TABLE (  partner\_name VARCHAR(100),  contract\_number VARCHAR(100),  booking\_start\_date DATE,  booking\_end\_date DATE)  AS $$  BEGIN  RETURN QUERY  SELECT p.organization, p.email, r.in\_data, r.out\_data FROM reservations r  JOIN partners p ON r.partner\_id = p.id  WHERE r.in\_data >= start\_date AND r.in\_data <= end\_date;  END;  $$ LANGUAGE plpgsql;  select \* from partner\_booking('2024-07-10','2024-07-20') |
| **9** | CREATE OR REPLACE FUNCTION client\_registrarion (fist\_data DATE, second\_data DATE)  RETURNS TABLE (id INT,  surname VARCHAR(100),  name VARCHAR(100),  patronymic VARCHAR(100),  registration\_data DATE,  passport VARCHAR(100),  birthday DATE,  address VARCHAR(200),  phone VARCHAR(30),  email VARCHAR(100))  AS $$  BEGIN  RETURN QUERY  SELECT \* FROM clients c  WHERE c.registration\_data between fist\_data and second\_data;  END;  $$ LANGUAGE plpgsql;  select \* from client\_registrarion('2024-01-01', '2024-07-08') |
| **10** | CREATE or replace FUNCTION get\_client\_info (sur VARCHAR(150), nam VARCHAR(150), patr VARCHAR(150))  RETURNS TABLE (room INT, check\_in\_data DATE, check\_out\_data DATE)  AS $$  BEGIN  RETURN QUERY  SELECT h.room\_number, h.check\_in\_data, h.check\_out\_data  FROM history h  LEFT JOIN clients c ON h.client\_id = c.id  WHERE h.client\_id = (SELECT id FROM clients  where surname = sur and name = nam and patronymic = patr);  END;  $$ LANGUAGE plpgsql;  select \* from get\_client\_info('Савельев','Илья','Дмитриевич'); |
| **11** | CREATE OR REPLACE FUNCTION concerned\_room (nomer INT, start\_data DATE, end\_data DATE)  RETURNS TABLE (  client text)  AS $$  BEGIN  RETURN QUERY  SELECT CONCAT(surname, ' ', name, ' ', patronymic) from clients c  JOIN history h ON c.id = h.client\_id  WHERE h.room\_number = nomer and h.check\_in\_data>=start\_data and h.check\_out\_data<=end\_data  group by CONCAT(surname, ' ', name, ' ', patronymic);  END;  $$ LANGUAGE plpgsql;  select \* from concerned\_room (1101, '2024-06-15', '2024-07-15') |
| **12** | CREATE OR REPLACE FUNCTION partner\_percentage(start\_date DATE, end\_date DATE)  RETURNS NUMERIC  AS  $$  DECLARE  total\_reservations INT;  partner\_reservations INT;  BEGIN  SELECT COUNT(\*) INTO total\_reservations  FROM reservations  WHERE in\_data >= start\_date AND out\_data <= end\_date;  SELECT COUNT(\*) INTO partner\_reservations  FROM reservations  WHERE in\_data >= start\_date AND out\_data <= end\_date  AND partner\_id IS NOT NULL;  IF total\_reservations = 0 THEN  RETURN 0;  ELSE  RETURN ROUND((partner\_reservations::NUMERIC / total\_reservations) \* 100,2);  END IF;  END;  $$  LANGUAGE PLPGSQL;  select \* from partner\_percentage('2024-07-01','2024-07-31') |
| **13** | CREATE OR REPLACE FUNCTION employee\_schedule (number INT)  RETURNS TABLE (  data DATE,  startt TIME,  endt TIME)  AS $$  BEGIN  RETURN QUERY  SELECT s.data, start\_time, end\_time FROM schedule s  WHERE employee\_id = number  ORDER BY data;  END;  $$ LANGUAGE plpgsql;  SELECT \* FROM employee\_schedule(1); |
| **14** | CREATE OR REPLACE FUNCTION booking\_by\_employee (sur VARCHAR(150), nam VARCHAR(150), patr VARCHAR(150))  RETURNS TABLE (  booking\_id INT,  client\_surname VARCHAR(200),  client\_name VARCHAR(200),  client\_patronymic VARCHAR(200),  partner VARCHAR(200),  corp INT,  room INT,  f\_data DATE,  s\_data DATE)  AS $$  BEGIN  RETURN QUERY  SELECT r.id, c.surname, c.name, c.patronymic, p.organization, r.corp\_number, r.room\_number, r.in\_data, r.out\_data FROM reservations r  LEFT JOIN clients c ON r.client\_id = c.id  LEFT JOIN partners p ON r.partner\_id = p.id  WHERE r.employee\_id = (SELECT id FROM employees  where surname = sur and name = nam and patronymic = patr);  END;  $$ LANGUAGE plpgsql;  SELECT \* FROM booking\_by\_employee('Белова','Анна','Сергеевна'); |
| **15** | CREATE OR REPLACE FUNCTION worked\_days(start\_data DATE, end\_data DATE)  RETURNS TABLE (surname VARCHAR(100),  name VARCHAR(100),  patronymic VARCHAR(100),  worked\_days BIGINT,  worked\_hours NUMERIC)  AS $$  BEGIN  RETURN QUERY  SELECT e.surname, e.name, e.patronymic, COUNT(DISTINCT s.data), SUM(EXTRACT(HOUR FROM (s.end\_time - s.start\_time))) AS worked\_hours  FROM employees e  JOIN schedule s ON e.id = s.employee\_id  WHERE s.data BETWEEN start\_data AND end\_data  GROUP BY e.surname, e.name, e.patronymic;  END;  $$ LANGUAGE plpgsql;  SELECT \* FROM worked\_days('2024-07-01', '2024-07-31'); |

|  |  |
| --- | --- |
| **Процедуры:** | |
| **1** | CREATE OR REPLACE PROCEDURE set\_schedule(IN employee\_id INT, IN data DATE, IN start\_time TIME, IN end\_time TIME)  LANGUAGE plpgsql AS $$  BEGIN  INSERT INTO schedule (id, employee\_id, data, start\_time, end\_time)  VALUES ((SELECT COALESCE(MAX(id), 0) + 1 FROM schedule), employee\_id, data, start\_time, end\_time);  END;  $$;  call set\_schedule(1,'2024-07-09','07:00:00', '16:00:00') |
| **2** | CREATE OR REPLACE PROCEDURE inc\_price(percent DECIMAL(5,2))  LANGUAGE plpgsql AS $$  BEGIN  UPDATE services SET price = price + (price \* percent / 100);  END;  $$;  call inc\_price(20) |

|  |  |
| --- | --- |
| **Триггеры:** | |
| **1** | create or replace function ins\_del\_reservations()  returns trigger as $$  begin  if TG\_OP = 'INSERT' then  UPDATE rooms  SET booking\_id = NEW.id  WHERE id = NEW.room\_number;  return new;  end if;  if TG\_OP = 'DELETE' then  UPDATE rooms  SET booking\_id = NULL  WHERE id = OLD.room\_number;  return new;  end if;  end;  $$ language plpgsql;  create or replace trigger before\_ins\_reservation  before insert on reservations  for each row  execute function ins\_del\_reservations();  create or replace trigger after\_del\_reservation  after delete on reservations  for each row  execute function ins\_del\_reservations(); |
| **2** | create or replace function update\_debt()  returns trigger as $$  begin  UPDATE history  SET debt = New.draft - NEW.payment  WHERE id = NEW.id;  return new;  end;  $$ language plpgsql;  create or replace trigger after\_insert\_history  after insert or update of payment,draft on history  for each row  execute function update\_debt(); |