Скрипты создания таблиц

```
CREATE TABLE IF NOT EXISTS users (
    user_id     BIGSERIAL PRIMARY KEY,
    last_name   VARCHAR(50) NOT NULL,
    first_name  VARCHAR(50) NOT NULL,
    middle_name VARCHAR(50),
    email       VARCHAR(512) NOT NULL,
    role        VARCHAR(50)
);
CREATE TABLE IF NOT EXISTS calendar (
    date        DATE PRIMARY KEY,
    type        VARCHAR(50) NOT NULL
);
CREATE TABLE IF NOT EXISTS meetings (
    meeting_id   BIGSERIAL PRIMARY KEY,
    title        VARCHAR(255) NOT NULL,
    date         DATE REFERENCES calendar(date),
    start_time   TIME,
    end_time     TIME,
    initiator_id BIGINT REFERENCES users(user_id),
    description  VARCHAR(7000),
    status       VARCHAR(50),
    url          VARCHAR(255)
);
CREATE TABLE IF NOT EXISTS requests (
    id           BIGSERIAL PRIMARY KEY,
    meeting_id     BIGINT REFERENCES meetings(meeting_id),
    initiator_id   BIGINT REFERENCES users(user_id),
    participant_id BIGINT REFERENCES users(user_id),
    status         VARCHAR(50)
);
CREATE TABLE IF NOT EXISTS meeting_participant (
    id             BIGSERIAL PRIMARY KEY,
    meeting_id     BIGINT REFERENCES meetings(meeting_id),
    participant_id BIGINT REFERENCES users(user_id)
);
```

--- Внесение данных в календарь

```
INSERT INTO calendar (date, type)
SELECT d,
       CASE
           WHEN EXTRACT(ISODOW FROM d) IN (6, 7) THEN 'HOLIDAY'
           ELSE 'WORKDAY'
       END AS type
FROM generate_series('2024-01-01'::date, '2024-12-31'::date, '1 day'::interval) AS d;
```

Аналитика

1) Распределение пользователей по ролям

```
create table  analytics.distribution_of_users (
        role varchar(50),
        count_users bigint
);

CREATE OR REPLACE PROCEDURE analytics.calculating_user_allocation ()
LANGUAGE plpgsql
AS $$
BEGIN
   DELETE FROM analytics.distribution_of_users;

   INSERT INTO analytics.distribution_of_users (role, count_users)
   SELECT
      role,
      count(user_id) as count_users
   FROM public.users
   GROUP BY role;
END;
$$;

call analytics.calculating_user_allocation ()
select * from analytics.distribution_of_users
```

2) Общее количество встреч по статусам

```
create table if not exists analytics.meeting_statuses(
        meeting_status varchar(50),
        count_statuses bigint
);

SELECT status, count(status) as count_statuses
FROM public.meetings
GROUP BY status
ORDER BY count_statuses DESC;

CREATE OR REPLACE PROCEDURE analytics.calculating_meeting_statuses ()
LANGUAGE plpgsql
AS $$
BEGIN
   DELETE FROM analytics.meeting_statuses;

   INSERT INTO analytics.meeting_statuses (meeting_status, count_statuses)
   SELECT
      status,
      count(status) as count_statuses
   FROM public.meetings
        GROUP BY status
        ORDER BY count_statuses DESC;
END;
$$;

call analytics.calculating_meeting_statuses ()
select * from analytics.meeting_statuses
```

3) Количество встреч рекрутеров

```
create table if not exists analytics.recruiter_interviews(
        first_name varchar(50),
        middle_name varchar(50),
        last_name varchar(50),
        count_interviews bigint
);

CREATE OR REPLACE PROCEDURE analytics.calculating_recruiter_interviews ()
LANGUAGE plpgsql
AS $$
BEGIN
   DELETE FROM analytics.recruiter_interviews;

   INSERT INTO analytics.recruiter_interviews (first_name, middle_name,last_name,count_interviews)
   SELECT
      first_name,
      middle_name,
            last_name,
            count(meeting_id) as count_interviews
   FROM public.users
        LEFT JOIN meetings ON user_id = initiator_id
        GROUP BY user_id
        ORDER BY count_interviews DESC;
END;
$$;

call analytics.calculating_recruiter_interviews ()
select * from analytics.recruiter_interviews
```

4) Процент успешных встреч пользователей

```
create table if not exists analytics.percentage_of_success(
   user_id bigint,
        first_name varchar(50),
        last_name varchar(50),
        middle_name varchar(50),
        meeting_ratio bigint
);

CREATE OR REPLACE PROCEDURE analytics.calculating_percentage_of_success ()
LANGUAGE plpgsql
AS $$
BEGIN
   DELETE FROM analytics.percentage_of_success;

   INSERT INTO analytics.percentage_of_success (user_id,first_name,last_name,middle_name,meeting_ratio)
   with count_confirmed as (
        SELECT
            u.user_id,
            count(me.meeting_id) as confirmed_count
        from public.users u
            RIGHT JOIN meeting_participant mep ON u.user_id = participant_id
            LEFT JOIN meetings me ON me.meeting_id = mep.meeting_id
            WHERE me.status = 'CONFIRMED'
            GROUP BY u.user_id)
        select u.user_id,
            u.first_name,
            u.last_name,
            u.middle_name,
            ((100* COALESCE(cc.confirmed_count, 1)) / count(me.meeting_id)) AS meeting_ratio
        from public.users u
            RIGHT JOIN meeting_participant mep ON u.user_id = participant_id
            LEFT JOIN meetings me ON me.meeting_id = mep.meeting_id
            LEFT JOIN count_confirmed cc ON u.user_id = cc.user_id
            GROUP BY u.user_id, cc.confirmed_count;
END;
$$;

call analytics.calculating_percentage_of_success ()
select * from analytics.percentage_of_success
```

5)Предстоящие встречи

```
create table if not exists analytics.future_meeting(
  user_id bigint,
      first_name varchar(50),
      last_name varchar(50),
      middle_name varchar(50),
     date_meeting date
);

CREATE OR REPLACE PROCEDURE analytics.calculating_future_meeting ()
LANGUAGE plpgsql
AS $$
BEGIN
  DELETE FROM analytics.future_meeting;

  INSERT INTO analytics.future_meeting (user_id,first_name,last_name,middle_name,date_meeting)
      select
      u.user_id,
      u.first_name,
      u.last_name,
      u.middle_name,
      me.date
from public.users u
RIGHT JOIN public.meeting_participant mp ON mp.participant_id = u.user_id
LEFT JOIN public.meetings me ON me.meeting_id = mp.meeting_id
      WHERE me.status not in ('TOOK_PLACE','NOT_HAPPENED') and me.date>current_date;

END;
$$;
call analytics.calculating_future_meeting ()
select * from analytics.future_meeting
```

6) Количество встреч по дням недели

```
BEGIN

  DELETE FROM analytics.meeting_week;

  INSERT INTO analytics.meeting_week (day_of_week,count_meeting)

  SELECT

    TO_CHAR(date,'Day') as day_of_week,

                count(meeting_id) as count_meeting
```

```
   FROM public.meetings
            GROUP BY day_of_week;
END;
```

7) Информация о встречах

```
create table if not exists analytics.general_info_meetings(
   meeting_id bigint,
            meeting_title varchar(255),
            meeting_date date,
            meeting_start_time timestamp,
            meeting_end_time timestamp,
            initiator_id bigint,
            initiator_name text,
            parcticipant_id bigint,
            partcicipant_name text,
            request_status varchar(50)
);



CREATE OR REPLACE PROCEDURE analytics.calculating_general_info_meetings ()
LANGUAGE plpgsql
AS $$
BEGIN
   DELETE FROM analytics.general_info_meetings;

   INSERT INTO analytics.general_info_meetings (meeting_id,
            meeting_title,
            meeting_date,
            meeting_start_time,
            meeting_end_time,
            initiator_id,
            initiator_name,
            parcticipant_id,
            partcicipant_name,
        request_status);
        SELECT
          m.meeting_id,
          m.title AS meeting_title,
          m.date AS meeting_date,
          m.start_time AS meeting_start_time,
          m.end_time AS meeting_end_time,
          u1.user_id AS initiator_id,
          u1.first_name ||' '|| u1.last_name AS initiator_name,
          u2.user_id AS participant_id,
          u2.first_name ||' '|| u2.last_name AS participant_name,
          r.status AS request_status
        FROM public.meetings m
        LEFT JOIN public.users u1 ON m.initiator_id = u1.user_id
        LEFT JOIN public.requests r ON m.meeting_id = r.meeting_id
        LEFT JOIN public.users u2 ON r.participant_id = u2.user_id;

END;
$$;
call analytics.calculating_general_info_meetings ()
select * from analytics.general_info_meetings
```