POSTGRESQL DEVELOPMENT GUIDELINE

Mục lục

Prepare environments

Cài đặt PostgreSQL

Bạn có thể bỏ qua phần này nếu đã cài đặt PostgreSQL.

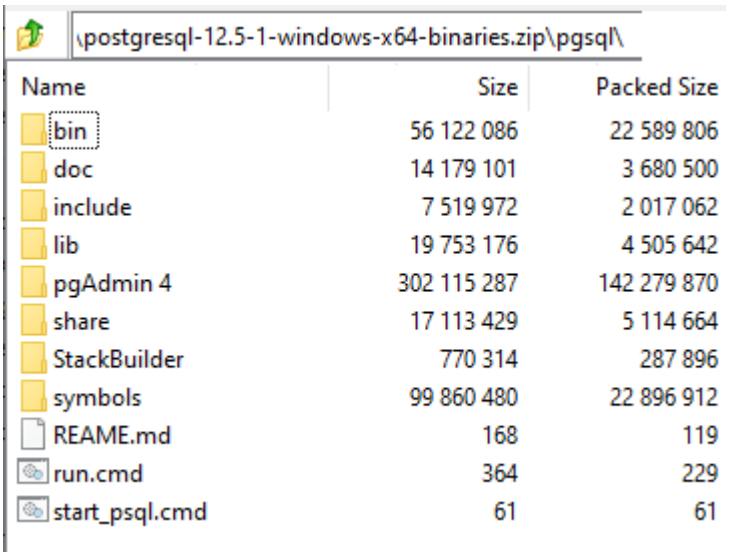
Tải gói binary này:

https://bit.ly/36WioU7

Gói này bao gồm bản postgresql gốc 12.5 được download tại:

https://www.enterprisedb.com/download-postgresql-binaries

và 2 scripts tôi tạo thêm: **`run.cmd`** và **`start_psql.cmd`**.

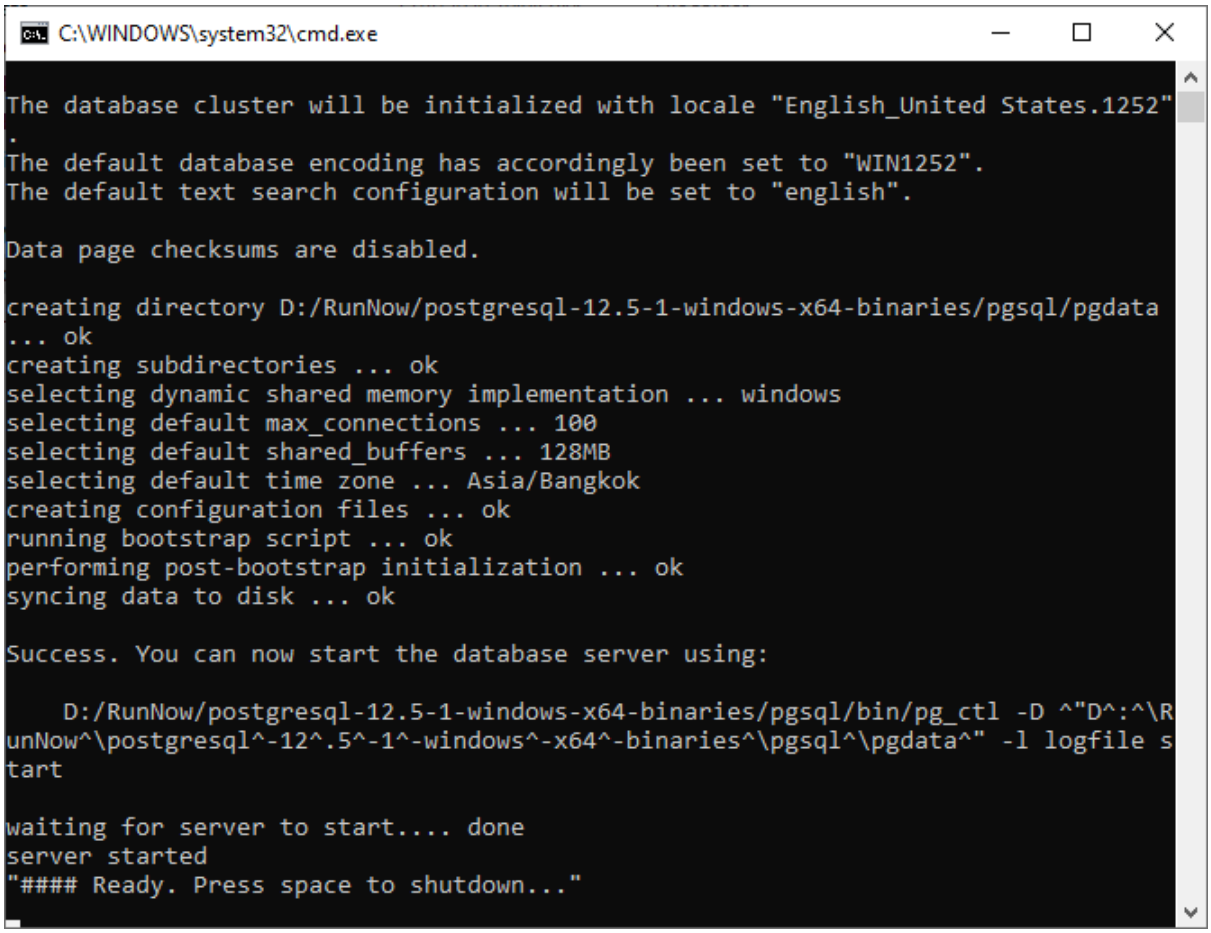


Sau khi tải về giải nén ra thư mục nào đó. Ví dụ giải nén ra thư mục D:\RunNow thì kết quả là có thư mục D:\RunNow\postgresql-12.5-1-windows-x64-binaries\pgsql.

Đứng trong thư mục này bạn khởi động PostgreSQL được cấu hình tại port 5439 bằng cách chạy script:

```
run.cmd
```

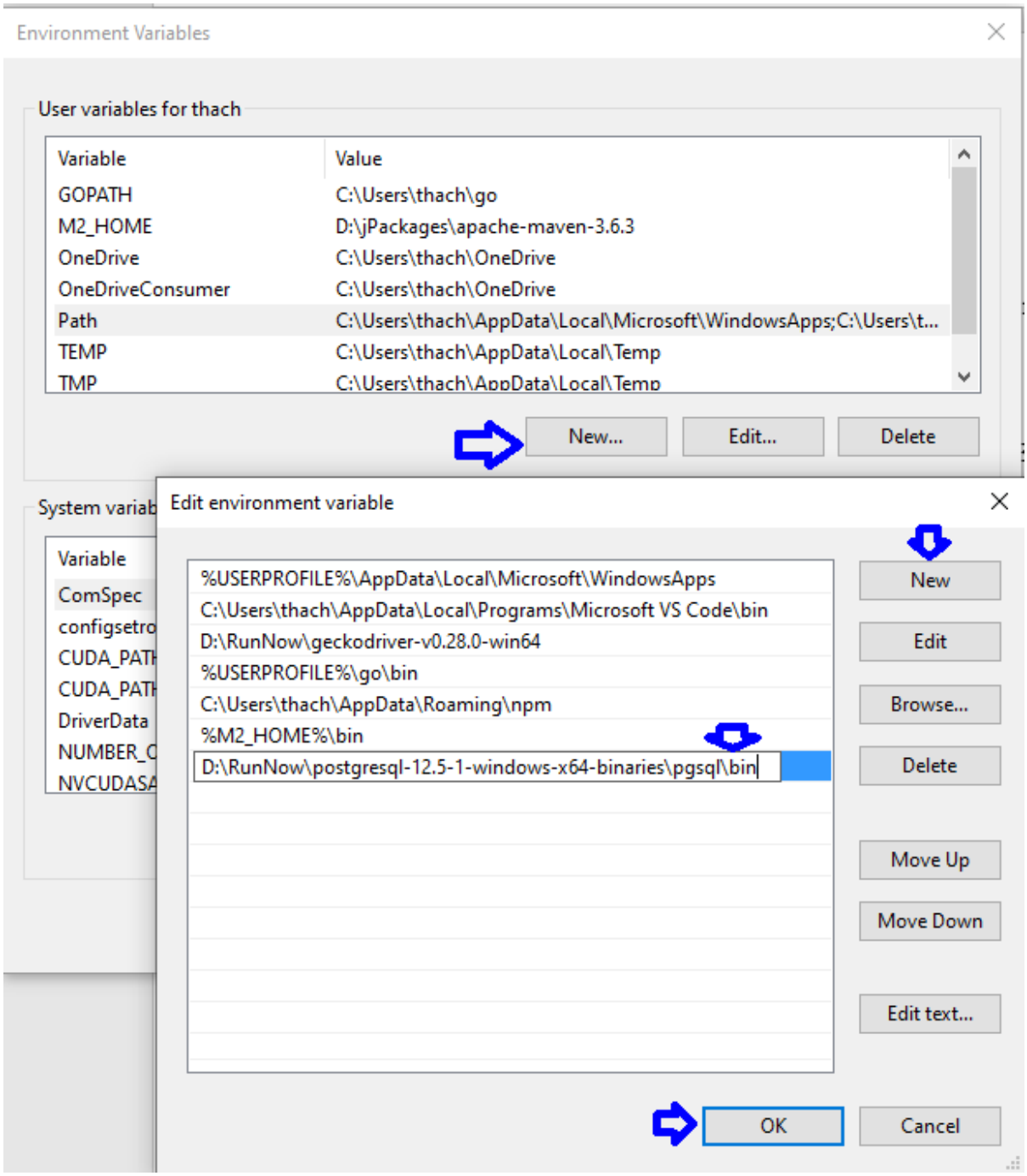
Kết quả là cửa sổ hiện ra bên dưới:



Sử dụng PostgreSQL

Bổ sung đường dẫn sau vào biến môi trường PATH:

D:\RunNow\postgresql-12.5-1-windows-x64-binaries\pgsql\bin

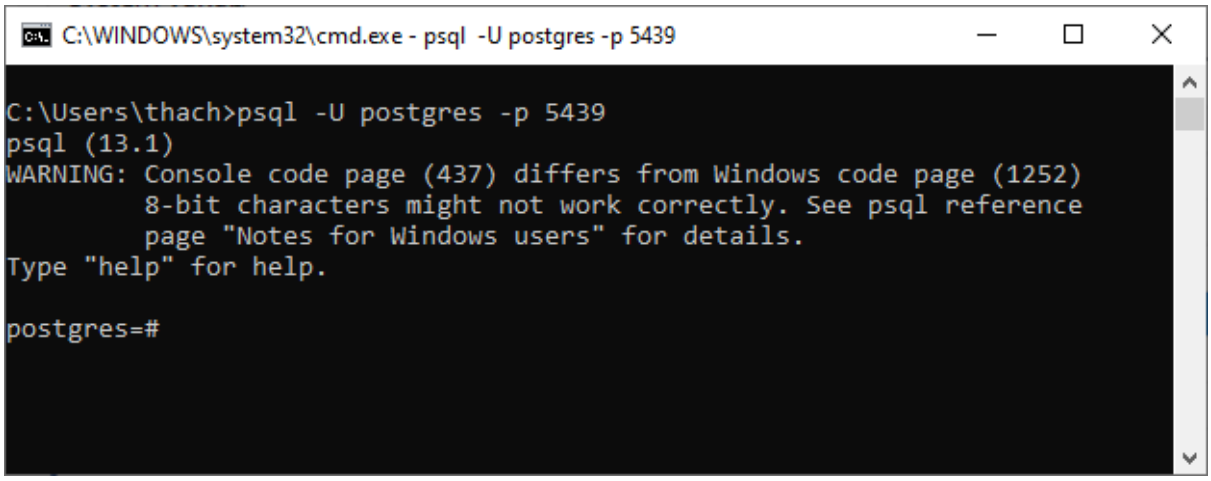


Kết nối server

Thực hiện lệnh sau để kết nối với PostgreSQL, port 5439 với user là "postgres"

```
psql -U postgres -p 5439
```

Kết quả sẽ ra dấu nhắc của postgres như bên dưới:



Bạn có thể thử vài lệnh sau cho quen:

\l (xuyệt trái và chữ L thường) để liệt kê (list) các database

\q để thoát dấu nhắc lệnh của PostgreSQL

Kết nối PostgreSQL ngay trong Ubuntu

```
sudo -u postgres psql
```

Tạo role và gán quyền

Mở kết nối tới server bằng lệnh sau:

```
psql -U postgres -p 5439
```

Thực hiện các lệnh bên dưới để trải nghiệm kỹ năng tạo database trong dấu nhắc lệnh của PostgreSQL:

```
CREATE DATABASE mydata;
CREATE USER mydata_user WITH PASSWORD 'p@ssw0rd';
GRANT CONNECT ON DATABASE mydata TO mydata_user;
GRANT ALL PRIVILEGES ON DATABASE "mydata" to mydata_user;
```

Mở cửa sổ cmd hỗ trợ UTF8

Tham khảo:

https://stackoverflow.com/questions/57131654/using-utf-8-encoding-chcp-65001-in-command-prompt-windows-powershell-window

Kết nối database icompset

Thử kết nối vào database **mydata** với username và password đã tạo bằng lệnh sau:

```
psql -U mydata_user -d mydata -h 127.0.0.1 -p 5439 -W
```

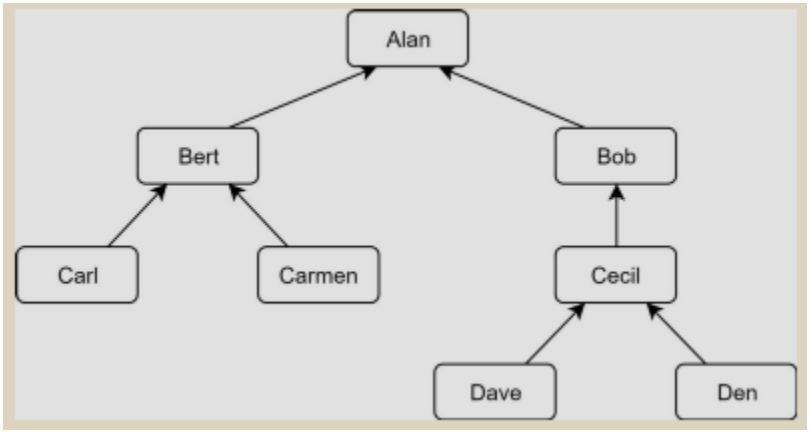
Gõ mật khẩu: p@ssw0rd

Nhấn Enter thì kết quả sẽ ra dấu nhắc lệnh như bên dưới:

```
C:\WINDOWS\system32\cmd.exe
psql (12.5)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

postgres=# psql -U mydata_user -d mydata -h 127.0.0.1 -p 5439 -W
postgres-# p@ssw0rd
postgres-#
```

Thử 2 câu lệnh sau để tạo ra table family để mô tả mối quan hệ như sau:




```
CREATE TABLE family (person text PRIMARY KEY, parent text
REFERENCES family);

INSERT INTO family VALUES ('Alan', NULL), ('Bert', 'Alan'),
('Bob', 'Alan'), ('Carl', 'Bert'), ('Carmen', 'Bert'),
('Cecil', 'Bob'), ('Dave', 'Cecil'), ('Den', 'Cecil');
```

Thực hiện 2 câu lệnh sau:

```
WITH RECURSIVE genealogy (bloodline, person, level) AS

(

SELECT person, person, 0 FROM family WHERE parent IS NULL
UNION ALL

SELECT g.bloodline || ' -> ' || f.person, f.person, g.level +
1

FROM family f, genealogy g WHERE f.parent = g.person

)
```

```
SELECT bloodline, level FROM genealogy;
```

```
          bloodline           | level
------------------------------+-------
 Alan                         |     0
```

```
 Alan -> Bert                 |     1
 Alan -> Bob                  |     1
 Alan -> Bert -> Carl         |     2
 Alan -> Bert -> Carmen       |     2
 Alan -> Bob -> Cecil         |     2
 Alan -> Bob -> Cecil -> Dave |     3
 Alan -> Bob -> Cecil -> Den  |     3
(8 rows)
```

Tham khảo cách tạo database và insert dữ liệu tại:

- https://github.com/PacktPublishing/Learning-PostgreSQL-10-Second-Edition/blob/master/Chapter03/building_blocks/schema.sql
- https://github.com/PacktPublishing/Learning-PostgreSQL-10-Second-Edition/blob/master/Chapter03/building_blocks/data.sql

Tạo database với user riêng

```
CREATE DATABASE db name;
CREATE USER db_user WITH PASSWORD 'Abc!123';
GRANT CONNECT ON DATABASE db name TO db user;
GRANT ALL PRIVILEGES ON DATABASE db_name to db_user;
```

Xóa database

```
REVOKE CONNECT ON DATABASE db_name FROM public;


SELECT pg_terminate_backend(pg_stat_activity.pid) FROM
pg_stat_activity WHERE pg_stat_activity.datname = 'db_name';
drop database db_name;
```

Tạo table

CREATE TABLE Step1

(

"Room type | Package" varchar(64),

"PR11028BB" float(19),

"PR11028BX" float(19),

"PR11028FB" float(19),

"PR11028FX" float(19),

"PR11031BB" float(19),

"PR11031BX" float(19),

"PR11031FB" float(19),

```
    "PR11031FX" float(19),
    "T2BB" int,
    "T2BX" int,
    "T2FB" int,
    "T2FX" int,
    "T2FXSFD" int,
    "PR11028BB_N" float(19),
    "PR11028BX_N" float(19),
    "PR11028FB_N" float(19),
    "PR11028FX_N" float(19),
    "PR11031BB_N" float(19),
    "PR11031BX_N" float(19),
    "PR11031FB_N" float(19),
    "PR11031FX_N" float(19),
    "T2BB_N" int,
    "T2BX_N" int,
    "T2FB_N" int,
    "T2FX_N" int,
    "T2FXSFD_N" int,
    "Hotel" varchar(128),
    "Guest" varchar(64),
    "Room" int,
    "checkIn" date,
    "checkOut" date,
    "updated" timestamp
);
```

CREATE UNIQUE INDEX CONCURRENTLY step1_unique ON step1("Room type | Package", "Hotel", "updated", "Guest", "Room", "checkIn", "checkOut");

ALTER TABLE step1 ADD CONSTRAINT step1_unique_cols UNIQUE USING INDEX step1_unique;

```
GRANT ALL PRIVILEGES ON TABLE Step1 TO db_user;
```

```
DROP TABLE if exists d_date;
CREATE TABLE d_date
(
  date_dim_id              INT NOT NULL,
  date_actual              DATE NOT NULL,
  epoch                    BIGINT NOT NULL,
  day_suffix               VARCHAR(4) NOT NULL,
  day_name                 VARCHAR(9) NOT NULL,
  day_of_week              INT NOT NULL,
  day_of_month             INT NOT NULL,
  day_of_quarter           INT NOT NULL,
  day_of_year              INT NOT NULL,
  week_of_month            INT NOT NULL,
  week_of_year             INT NOT NULL,
  week_of_year_iso         CHAR(10) NOT NULL,
  month_actual             INT NOT NULL,
  month_name               VARCHAR(9) NOT NULL,
  month_name_abbreviated   CHAR(3) NOT NULL,
  quarter_actual           INT NOT NULL,
  quarter_name             VARCHAR(9) NOT NULL,
  year_actual              INT NOT NULL,
  first_day_of_week        DATE NOT NULL,
  last_day_of_week         DATE NOT NULL,
  first_day_of_month       DATE NOT NULL,
  last_day_of_month        DATE NOT NULL,
  first_day_of_quarter     DATE NOT NULL,
  last_day_of_quarter      DATE NOT NULL,
  first_day_of_year        DATE NOT NULL,
  last_day_of_year         DATE NOT NULL,
  mmyyyy                   CHAR(6) NOT NULL,
  mmddyyyy                 CHAR(10) NOT NULL,
  weekend_indr             BOOLEAN NOT NULL
);

ALTER TABLE d_date ADD CONSTRAINT d_date_date_dim_id_pk PRIMARY KEY (date_dim_id);

CREATE INDEX d_date_date_actual_idx
  ON d_date(date_actual);

COMMIT;

INSERT INTO d_date
SELECT TO_CHAR(datum, 'yyyymmdd')::INT AS date_dim_id,
       datum AS date_actual,
       EXTRACT(EPOCH FROM datum) AS epoch,
```

```
       TO_CHAR(datum, 'fmDDth') AS day_suffix,
       TO_CHAR(datum, 'TMDay') AS day_name,
       EXTRACT(ISODOW FROM datum) AS day_of_week,
       EXTRACT(DAY FROM datum) AS day_of_month,
       datum - DATE_TRUNC('quarter', datum)::DATE + 1 AS day_of_quarter,
       EXTRACT(DOY FROM datum) AS day_of_year,
       TO_CHAR(datum, 'W')::INT AS week_of_month,
       EXTRACT(WEEK FROM datum) AS week_of_year,
       EXTRACT(ISOYEAR FROM datum) || TO_CHAR(datum, '"-W"IW-') || EXTRACT(ISODOW FROM
datum) AS week_of_year_iso,
       EXTRACT(MONTH FROM datum) AS month_actual,
       TO_CHAR(datum, 'TMMonth') AS month_name,
       TO_CHAR(datum, 'Mon') AS month_name_abbreviated,
       EXTRACT(QUARTER FROM datum) AS quarter_actual,
       CASE
           WHEN EXTRACT(QUARTER FROM datum) = 1 THEN 'First'
           WHEN EXTRACT(QUARTER FROM datum) = 2 THEN 'Second'
           WHEN EXTRACT(QUARTER FROM datum) = 3 THEN 'Third'
           WHEN EXTRACT(QUARTER FROM datum) = 4 THEN 'Fourth'
           END AS quarter_name,
       EXTRACT(YEAR FROM datum) AS year_actual,
       datum + (1 - EXTRACT(ISODOW FROM datum))::INT AS first_day_of_week,
       datum + (7 - EXTRACT(ISODOW FROM datum))::INT AS last_day_of_week,
       datum + (1 - EXTRACT(DAY FROM datum))::INT AS first_day_of_month,
       (DATE_TRUNC('MONTH', datum) + INTERVAL '1 MONTH - 1 day')::DATE AS
last_day_of_month,
       DATE_TRUNC('quarter', datum)::DATE AS first_day_of_quarter,
       (DATE_TRUNC('quarter', datum) + INTERVAL '3 MONTH - 1 day')::DATE AS
last_day_of_quarter,
       TO_DATE(EXTRACT(YEAR FROM datum) || '-01-01', 'YYYY-MM-DD') AS first_day_of_year,
       TO_DATE(EXTRACT(YEAR FROM datum) || '-12-31', 'YYYY-MM-DD') AS last_day_of_year,
       TO_CHAR(datum, 'mmyyyy') AS mmyyyy,
       TO_CHAR(datum, 'mmddyyyy') AS mmddyyyy,
       CASE
           WHEN EXTRACT(ISODOW FROM datum) IN (6, 7) THEN TRUE
           ELSE FALSE
           END AS weekend_indr
FROM (SELECT '2020-01-01'::DATE + SEQUENCE.DAY AS datum
      FROM GENERATE_SERIES(0, 29219) AS SEQUENCE (DAY)
      GROUP BY SEQUENCE.DAY) DQ
ORDER BY 1;

COMMIT;
```

Store Procedure:

```
CREATE OR REPLACE PROCEDURE dwh.create_dim_date()
      LANGUAGE sql
AS $procedure$
      BEGIN
```

```
            DROP d_date if exists d_date;
            CREATE TABLE d_date
            (
              date_dim_id              INT NOT NULL,
              date_actual              DATE NOT NULL,
              epoch                    BIGINT NOT NULL,
              day_suffix               VARCHAR(4) NOT NULL,
              day_name                 VARCHAR(9) NOT NULL,
              day_of_week              INT NOT NULL,
              day_of_month             INT NOT NULL,
              day_of_quarter           INT NOT NULL,
              day_of_year              INT NOT NULL,
              week_of_month            INT NOT NULL,
              week_of_year             INT NOT NULL,
              week_of_year_iso         CHAR(10) NOT NULL,
              month_actual             INT NOT NULL,
              month_name               VARCHAR(9) NOT NULL,
              month_name_abbreviated   CHAR(3) NOT NULL,
              quarter_actual           INT NOT NULL,
              quarter_name             VARCHAR(9) NOT NULL,
              year_actual              INT NOT NULL,
              first_day_of_week        DATE NOT NULL,
              last_day_of_week         DATE NOT NULL,
              first_day_of_month       DATE NOT NULL,
              last_day_of_month        DATE NOT NULL,
              first_day_of_quarter     DATE NOT NULL,
              last_day_of_quarter      DATE NOT NULL,
              first_day_of_year        DATE NOT NULL,
              last_day_of_year         DATE NOT NULL,
              mmyyyy                   CHAR(6) NOT NULL,
              mmddyyyy                 CHAR(10) NOT NULL,
              weekend_indr             BOOLEAN NOT NULL
            );


            ALTER TABLE d_date ADD CONSTRAINT d_date_date_dim_id_pk PRIMARY KEY
(date_dim_id);


            CREATE INDEX d_date_date_actual_idx
              ON d_date(date_actual);


            COMMIT;


            INSERT INTO d_date
            SELECT TO_CHAR(datum, 'yyyymmdd')::INT AS date_dim_id,
                   datum AS date_actual,
                   EXTRACT(EPOCH FROM datum) AS epoch,
                   TO_CHAR(datum, 'fmDDth') AS day_suffix,
                   TO_CHAR(datum, 'TMDay') AS day_name,
                   EXTRACT(ISODOW FROM datum) AS day_of_week,
                   EXTRACT(DAY FROM datum) AS day_of_month,
                   datum - DATE_TRUNC('quarter', datum)::DATE + 1 AS
day_of_quarter,
                   EXTRACT(DOY FROM datum) AS day_of_year,
                   TO_CHAR(datum, 'W')::INT AS week_of_month,
                   EXTRACT(WEEK FROM datum) AS week_of_year,
                   EXTRACT(ISOYEAR FROM datum) || TO_CHAR(datum, '"-W"IW-') ||
EXTRACT(ISODOW FROM datum) AS week_of_year_iso,
```

```
                    EXTRACT(MONTH FROM datum) AS month_actual,
                    TO_CHAR(datum, 'TMMonth') AS month_name,
                    TO_CHAR(datum, 'Mon') AS month_name_abbreviated,
                    EXTRACT(QUARTER FROM datum) AS quarter_actual,
                    CASE
                        WHEN EXTRACT(QUARTER FROM datum) = 1 THEN 'First'
                        WHEN EXTRACT(QUARTER FROM datum) = 2 THEN 'Second'
                        WHEN EXTRACT(QUARTER FROM datum) = 3 THEN 'Third'
                        WHEN EXTRACT(QUARTER FROM datum) = 4 THEN 'Fourth'
                        END AS quarter_name,
                    EXTRACT(YEAR FROM datum) AS year_actual,
                    datum + (1 - EXTRACT(ISODOW FROM datum))::INT AS
first_day_of_week,
                    datum + (7 - EXTRACT(ISODOW FROM datum))::INT AS
last_day_of_week,
                    datum + (1 - EXTRACT(DAY FROM datum))::INT AS
first_day_of_month,
                    (DATE_TRUNC('MONTH', datum) + INTERVAL '1 MONTH - 1
day')::DATE AS last_day_of_month,
                    DATE_TRUNC('quarter', datum)::DATE AS first_day_of_quarter,
                    (DATE_TRUNC('quarter', datum) + INTERVAL '3 MONTH - 1
day')::DATE AS last_day_of_quarter,
                    TO_DATE(EXTRACT(YEAR FROM datum) || '-01-01', 'YYYY-MM-DD')
AS first_day_of_year,
                    TO_DATE(EXTRACT(YEAR FROM datum) || '-12-31', 'YYYY-MM-DD')
AS last_day_of_year,
                    TO_CHAR(datum, 'mmyyyy') AS mmyyyy,
                    TO_CHAR(datum, 'mmddyyyy') AS mmddyyyy,
                    CASE
                        WHEN EXTRACT(ISODOW FROM datum) IN (6, 7) THEN TRUE
                        ELSE FALSE
                        END AS weekend_indr
            FROM (SELECT '2020-01-01'::DATE + SEQUENCE.DAY AS datum
                  FROM GENERATE_SERIES(0, 29219) AS SEQUENCE (DAY)
                  GROUP BY SEQUENCE.DAY) DQ
            ORDER BY 1;


            COMMIT;

      END;
$procedure$
```

Simple Function:

```
CREATE OR REPLACE FUNCTION public.my_sum(integer, integer)
 RETURNS integer
 LANGUAGE sql
AS $function$
SELECT $1 + $2;
$function$
;
```

Refer:

https://duffn.medium.com/creating-a-date-dimension-table-in-postgresql-af3f8e2941ac

https://www.nuwavesolutions.com/simple-hierarchical-dimensions-html/

https://www.nuwavesolutions.com/ragged_hierarchical_dimensions/

https://wiki.postgresql.org/images/3/38/PGDay2009-EN-Datawarehousing_with_PostgreSQL.pdf