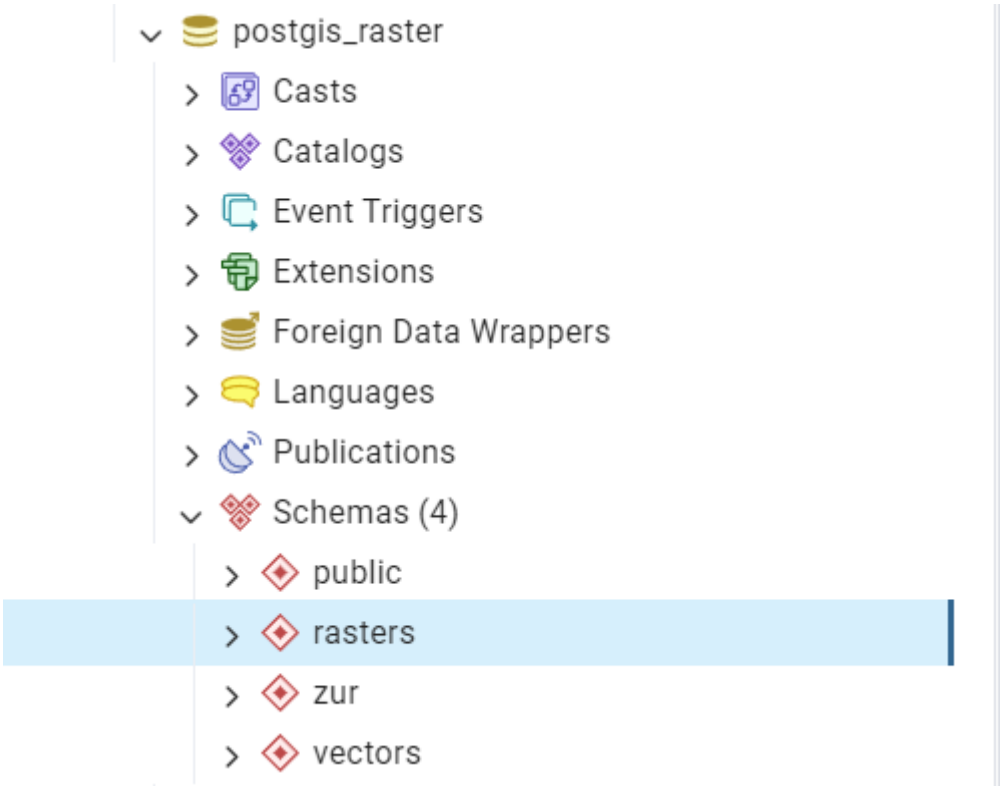
1. Załadowanie kopii bazy danych

```
C:\Users\kamaz>pg_restore -h localhost -U postgres -d postgis_raster -v "D:\Studia\Semestr 7\Bazy Danych Przestrzennych\
lab6-7\PostGIS raster - dane\PostGIS raster - dane\postgis_raster.backup"
pg_restore: warning: restoring tables WITH OIDS is not supported anymore
pg_restore: warning: restoring tables WITH OIDS is not supported anymore
pg_restore: connecting to database for restore
Password:
```



2. Wczytanie danych rastrowych

```
C:\Users\kamaz>D:\PostgreSQL\16\bin\raster2pgsql.exe -s 3763 -N -32767 -t 100x100 -I -C -M "D:\Studia\Semestr 7\Bazy Dan
ych Przestrzennych\lab6-7\PostGIS raster - dane\PostGIS raster - dane\srtm_1arc_v3.tif" rasters.dem | psql -U postgres -
d postgis_raster
Processing 1/1: D:\Studia\Semestr 7\Bazy Danych Przestrzennych\lab6-7\PostGIS raster - dane\PostGIS raster - dane\srtm_1
arc_v3.tif
Password for user postgres:
```

```
C:\Users\kamaz>D:\PostgreSQL\16\bin\raster2pgsql.exe -s 3763 -N -32767 -t 128x128 -I -C -M "D:\Studia\Semestr 7\Bazy Dan
ych Przestrzennych\lab6-7\PostGIS raster - dane\PostGIS raster - dane\Landsat8_L1TP_RGBN.tif" rasters.landsat8 | psql -U
 postgres -d postgis_raster
Processing 1/1: D:\Studia\Semestr 7\Bazy Danych Przestrzennych\lab6-7\PostGIS raster - dane\PostGIS raster - dane\Landsa
t8_L1TP_RGBN.tif
Password for user postgres:
```

3. ST_Intersects - tworzy tabelę z rastrami nakładającymi się na geometrię

```sql
-- ST_Intersects
CREATE TABLE zur.intersects AS
SELECT a.rast, b.municipality
FROM rasters.dem AS a, vectors.porto_parishes AS b
WHERE ST_Intersects(a.rast, b.geom) AND b.municipality ILIKE 'porto';

alter table zur.intersects
add column rid SERIAL PRIMARY KEY;

CREATE INDEX idx_intersects_rast_gist ON zur.intersects
USING gist (ST_ConvexHull(rast));
```

4. ST_Clip - tworzy raster przycięty do geometrii

```sql
-- ST_Clip
CREATE TABLE zur.clip AS
SELECT ST_Clip(a.rast, b.geom, true), b.municipality
FROM rasters.dem AS a, vectors.porto_parishes AS b
WHERE ST_Intersects(a.rast, b.geom) AND b.municipality ILIKE 'porto';
```

5. ST_Union - tworzy pojedynczy raster z wielu kafelków

```sql
-- ST_Union
CREATE TABLE zur.union AS
SELECT ST_Union(ST_Clip(a.rast, b.geom, true))
FROM rasters.dem AS a, vectors.porto_parishes AS b
WHERE b.municipality ILIKE 'porto' AND ST_Intersects(b.geom, a.rast);
```

6. ST_AsRaster - tworzy raster na podstawie geometrii

```sql
-- ST_AsRaster
CREATE TABLE zur.porto_parishes AS
WITH r AS (SELECT rast FROM rasters.dem LIMIT 1)
SELECT ST_AsRaster(a.geom, r.rast, '8BUI', a.id, -32767) AS rast
FROM vectors.porto_parishes AS a, r
WHERE a.municipality ILIKE 'porto';
```

7. ST_Union - łączy rastry w jeden

```sql
-- ST_Union
DROP TABLE zur.porto_parishes;

CREATE TABLE zur.porto_parishes AS
WITH r AS (SELECT rast FROM rasters.dem LIMIT 1)
SELECT ST_Union(ST_AsRaster(a.geom, r.rast, '8BUI', a.id, -32767)) AS rast
FROM vectors.porto_parishes AS a, r
WHERE a.municipality ILIKE 'porto';
```

8. ST_Tile - rozdziela jeden raster na kafelki

```sql
-- ST_Tile
DROP TABLE zur.porto_parishes;

CREATE TABLE zur.porto_parishes AS
WITH r AS (SELECT rast FROM rasters.dem LIMIT 1)
SELECT st_tile(st_union(ST_AsRaster(a.geom,r.rast,'8BUI',a.id,-32767)),128,128,true,-32767) AS rast
FROM vectors.porto_parishes AS a, r
WHERE a.municipality ilike 'porto';
```

9. ST_Intersection - wyodrębnia geometrię i wartość pikseli z rastra w obszarze, gdzie przecina się on z geometrią wektora

```sql
-- ST_Intersection
CREATE TABLE zur.intersection AS
SELECT
    a.rid,
    (ST_Intersection(b.geom, a.rast)).geom AS geom,
    (ST_Intersection(b.geom, a.rast)).val AS val
FROM zur.landsat8 AS a, vectors.porto_parishes AS b
WHERE b.parish ILIKE 'paranhos' AND ST_Intersects(b.geom, a.rast);
```

10. ST_DumpAsPolygons - konwertuje raster na zestaw wektorów (poligonów)

```sql
-- ST_DumpAsPolygons
CREATE TABLE zur.dumppolygons AS
SELECT
    a.rid,
    (ST_DumpAsPolygons(ST_Clip(a.rast, b.geom))).geom AS geom,
    (ST_DumpAsPolygons(ST_Clip(a.rast, b.geom))).val AS val
FROM zur.landsat8 AS a, vectors.porto_parishes AS b
WHERE b.parish ILIKE 'paranhos' AND ST_Intersects(b.geom, a.rast);
```

11. ST_Band - wyodrębnia wskazane pasmo z rastra

```sql
-- ST_Band
CREATE TABLE zur.landsat_nir AS
SELECT rid, ST_Band(rast, 4) AS rast
FROM zur.landsat8;
```

12. ST_Clip - przycina raster do geometrii wektora

```sql
-- ST_Clip
CREATE TABLE zur.paranhos_dem AS
SELECT a.rid, ST_Clip(a.rast, b.geom, true) AS rast
FROM rasters.dem AS a, vectors.porto_parishes AS b
WHERE b.parish ILIKE 'paranhos' AND ST_Intersects(b.geom, a.rast);
```

13. ST_Slope - oblicza nachylenie każdego piksela w rastrze (jako procent)

```sql
-- ST_Slope
CREATE TABLE zur.paranhos_slope AS
SELECT a.rid, ST_Slope(a.rast, 1, '32BF', 'PERCENTAGE') AS rast
FROM zur.paranhos_dem AS a;
```

14. ST_Reclass - reklasyfikuje wartości rastra na klastry

```sql
-- ST_Reclass
CREATE TABLE zur.paranhos_slope_reclass AS
SELECT a.rid, ST_Reclass(a.rast, 1, ']0-15]:1, (15-30]:2, (30-9999]:3', '32BF', 0)
FROM zur.paranhos_slope AS a;
```

15. ST_SummaryStats - generuje statystyki (min, max, średnia, suma, liczba pikseli) dla każdego rekordu w tabeli

```sql
-- ST_SummaryStats
SELECT ST_SummaryStats(a.rast) AS stats
FROM zur.paranhos_dem AS a;

-- ST_SummaryStats oraz ST_Union
SELECT ST_SummaryStats(ST_Union(a.rast)) AS stats
FROM zur.paranhos_dem AS a;

-- ST_SummaryStats z lepszą kontrolą złożonego typu danych
WITH t AS (
    SELECT ST_SummaryStats(ST_Union(a.rast)) AS stats
    FROM zur.paranhos_dem AS a
)
SELECT (stats).min, (stats).max, (stats).mean FROM t;

-- ST_SummaryStats z GROUP BY
WITH t AS (
    SELECT b.parish AS parish, ST_SummaryStats(ST_Union(ST_Clip(a.rast, b.geom, t
    FROM rasters.dem AS a, vectors.porto_parishes AS b
    WHERE b.municipality ILIKE 'porto' AND ST_Intersects(b.geom, a.rast)
    GROUP BY b.parish
)
SELECT parish, (stats).min, (stats).max, (stats).mean FROM t;
```

16. ST_Value - wyciąga wartość piksela z rastra na podstawie punktu

```sql
-- ST_Value
SELECT b.name, ST_Value(a.rast, (ST_Dump(b.geom)).geom) AS pixel_value
FROM rasters.dem AS a, vectors.places AS b
WHERE ST_Intersects(a.rast, b.geom)
ORDER BY b.name;
```

17. ST_TPI - oblicza indeks pozycji topograficznej (TPI) dla każdego piksela w rastrze

```
-- ST_TPI
CREATE TABLE zur.tpi30 AS
SELECT ST_TPI(a.rast, 1) AS rast
FROM rasters.dem AS a;

CREATE INDEX idx_tpi30_rast_gist ON zur.tpi30
USING gist (ST_ConvexHull(rast));

SELECT AddRasterConstraints('zur'::name, 'tpi30'::name, 'rast'::name);
```

```
-- ST_TPI i ST_Intersects
CREATE TABLE zur.tpi30_porto AS
SELECT ST_TPI(a.rast, 1) AS rast
FROM rasters.dem AS a, vectors.porto_parishes AS b
WHERE ST_Intersects(a.rast, b.geom) AND b.municipality ILIKE 'porto';
```

18. Wyrażenie algebry map - oblicza wskaźnik NDVI dla obrazów satelitarnych

```
-- Wyrażenie algebry map
CREATE TABLE zur.porto_ndvi AS
WITH r AS (
    SELECT a.rid, ST_Clip(a.rast, b.geom, true) AS rast
    FROM rasters.landsat8 AS a, vectors.porto_parishes AS b
    WHERE b.municipality ILIKE 'porto' AND ST_Intersects(b.geom, a.rast)
)
SELECT r.rid, ST_MapAlgebra(r.rast, 1, r.rast, 4,
    '([rast2.val] - [rast1.val]) / ([rast2.val] + [rast1.val])::float', '32BF') A
FROM r;

CREATE INDEX idx_porto_ndvi_rast_gist ON zur.porto_ndvi
USING gist (ST_ConvexHull(rast));SELECT AddRasterConstraints('schema_name'::name,
'porto_ndvi'::name,'rast'::name);


SELECT AddRasterConstraints('zur'::name,'porto_ndvi'::name,'rast'::name);
```

19. Funkcja zwrotna

```
-- Funkcja zwrotna
create or replace function zur.ndvi(
    value double precision [] [] [],
    pos integer [][],
    VARIADIC userargs text []
)
RETURNS double precision AS
$$
BEGIN
    --RAISE NOTICE 'Pixel Value: %', value [1][1][1];-->For debug purposes
    RETURN (value [2][1][1] - value [1][1][1])/(value [2][1][1]+value
[1][1][1]); --> NDVI calculation!
END;
$$
LANGUAGE 'plpgsql' IMMUTABLE COST 1000;

CREATE TABLE zur.porto_ndvi2 AS
WITH r AS (
    SELECT a.rid,ST_Clip(a.rast, b.geom,true) AS rast
    FROM rasters.landsat8 AS a, vectors.porto_parishes AS b
    WHERE b.municipality ilike 'porto' and ST_Intersects(b.geom,a.rast)
)
SELECT
    r.rid,ST_MapAlgebra(
        r.rast, ARRAY[1,4],
        'zur.ndvi(double precision[],
integer[],text[])'::regprocedure, --> This is the function!
'32BF'::text
    ) AS rast
FROM r;
```

20. ST_AsTiff - generuje GeoTIFF z rastra NDVI

```
-- ST_AsTiff
SELECT ST_AsTiff(ST_Union(rast))
FROM zur.porto_ndvi;
```

21. ST_AsGDALRaster

```
-- ST_AsGDALRaster
SELECT ST_AsGDALRaster(ST_Union(rast), 'GTiff', ARRAY['COMPRESS=DEFLATE',
'PREDICTOR=2', 'PZLEVEL=9'])
FROM zur.porto_ndvi;
```

22. Large Objects

```sql
-- lo
CREATE TABLE tmp_out AS
SELECT lo_from_bytea(0, ST_AsGDALRaster(ST_Union(rast), 'GTiff', ARRAY['COMPRESS=DEFLATE', 'PREDICTOR=2', 'ZLEVEL=9'])) AS loid
FROM zur.porto_ndvi;

SELECT lo_export(loid, 'D:\Studia\Semestr 7\Bazy Danych Przestrzennych\lab6-7\porto_ndvi.tif') FROM tmp_out;
```