PG Tricks

Nordic PGDay 2026 - Helsinki

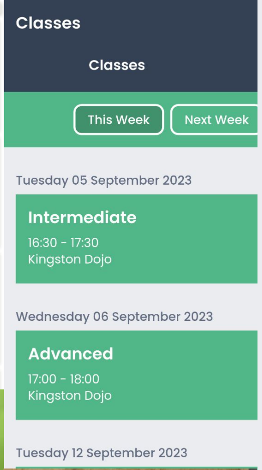
Chris Ellis - @intrbiz@bergamot.social

chris@nexteam.co.uk https://nexteam.co.uk

Hello!

- I’m Chris
  - IT jack of all trades, studied Electronic Engineering
  - These days, mostly a technical architect
  - Spend most of my time building apps on top of PostgreSQL
- Been using PostgreSQL for about ~20 years
- Worked on various PostgreSQL and IoT projects
- Head Of Technology - Nexteam
  - We help small and big companies with technology problems
  - I can help support you using PostgreSQL

2023
PGDay UK
2023
PGDay UK


Classes

Classes

This Week | Next Week

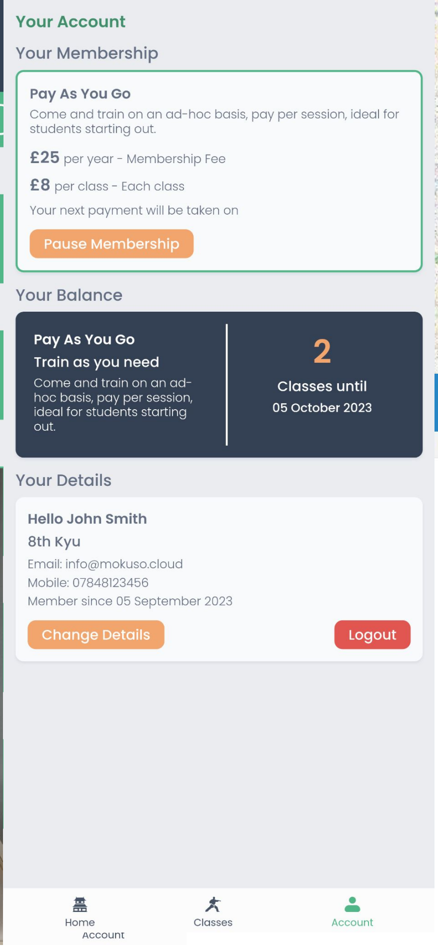
Tuesday 05 September 2023

Intermediate
16:30 - 17:30
Kingston Dojo

Wednesday 06 September 2023

Advanced
17:00 - 18:00
Kingston Dojo

Tuesday 12 September 2023


Your Account

Your Membership

Pay As You Go
Come and train on an ad-hoc basis, pay per session, ideal for students starting out.

£25 per year - Membership Fee

£8 per class - Each class

Your next payment will be taken on

Pause Membership

Your Balance

Pay As You Go
Train as you need
Come and train on an ad-hoc basis, pay per session, ideal for students starting out.

2
Classes until
05 October 2023

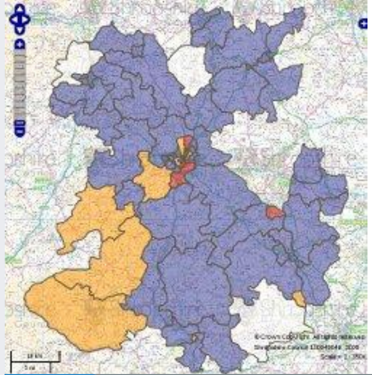
Your Details

Hello John Smith
8th Kyu
Email: info@mokuso.cloud
Mobile: 07848123456
Member since 05 September 2023

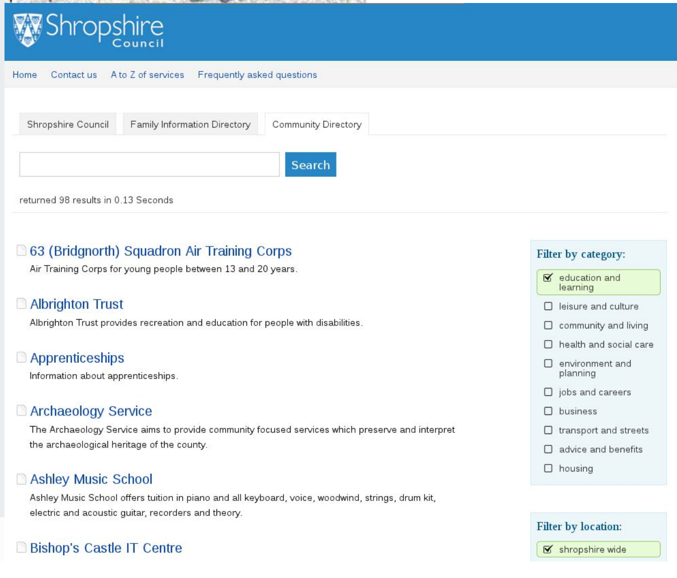
Change Details | Logout

Home | Classes | Account

Account



NEXTEAM


Shropshire Council

Home | Contact us | A to Z of services | Frequently asked questions

Shropshire Council | Family Information Directory | Community Directory

Search

returned 98 results in 0.13 Seconds

63 (Bridgnorth) Squadron Air Training Corps
Air Training Corps for young people between 13 and 20 years.

Albrighton Trust
Albrighton Trust provides recreation and education for people with disabilities.

Apprenticeships
Information about apprenticeships.

Archaeology Service
The Archaeology Service aims to provide community focused services which preserve and interpret the archaeological heritage of the county.

Ashley Music School
Ashley Music School offers tuition in piano and all keyboard, voice, woodwind, strings, drum kit, electric and acoustic guitar, recorders and theory.

Bishop's Castle IT Centre

Filter by category:
- [✓] education and learning
- [ ] leisure and culture
- [ ] community and living
- [ ] health and social care
- [ ] environment and planning
- [ ] jobs and careers
- [ ] business
- [ ] transport and streets
- [ ] advice and benefits
- [ ] housing

Filter by location:
- [✓] shropshire wide

chris@nexteam.co.uk

https://nexteam.co.uk

SELECT * FROM audience WHERE ...

Just Use PostgreSQL?

Local Dev

Feature Branches

```sql
CREATE DATABASE app_ft_plan_addons
  TEMPLATE app_db
  OWNER app_user;
```

Testing Queries

```
SET enable_seqscan = off;

EXPLAIN [ANALYSE] ...;
```

Data Types

UUID

```
CREATE TABLE club (
   id         UUID       PRIMARY KEY,
   tenant_id  UUID       NOT NULL
   ...
);
```

UUID - Information Inside

72a500f1-211f-7001-81c3-ed36b0748e0f

Time Random

000139a9-7064-8064-8000-9fba0fd0f203

Time Tenant Tag & Seq Random

Range Types

```
CREATE TABLE event (
  id       UUID         PRIMARY KEY,
  venue    UUID         NOT NULL,
  starts   TIMESTAMPTZ  NOT NULL,
  ends     TIMESTAMPTZ  NOT NULL
);
```

Range Types

```
SELECT *
FROM event
WHERE tstzrange(start, end)
          && tstzrange($1, $2);
```

Range Types

```
ALTER TABLE event
ADD COLUMN at TSTZRANGE
   GENERATED ALWAYS AS
      (tstzrange(starts, ends))
   STORED;
```

Range Types

```
CREATE EXTENSION btree_gist;

ALTER TABLE event
ADD CONSTRAINT no_event_overlaps
  EXCLUDE USING GIST
   (venue WITH =, at WITH &&);
```

Range Types

```
CREATE TABLE event (
  id          UUID                PRIMARY KEY,
  ...
  age_range   INT4RANGE
);
```

Range Types

```
SELECT *
FROM event
WHERE ...
  AND age_group @> 13;
```

Arrays - Roll Ups

```
CREATE TABLE daily_reading (
  meter_id       UUID        NOT NULL,
  day            DATE        NOT NULL,
  energy         BIGINT              ,
  energy_profile BIGINT[]            ,
  PRIMARY KEY (meter_id, day)
);
```

Arrays - Roll Ups

t_xmin	t_xmax	t_cid	t_xvac	t_ctid	t_infomask 2	t_infomask	t_hoff
4	4	4	4	6	2	2	1

24 bytes

device_id	read_at	temperature	light
16	8	4	4

32 bytes

```
{
  from: "01902600666"
  transcript: [
    "Hey, we've had a problem."
    "We've had a Main B bus undervolt",
    "We got a Main bus A undervolt, now, too...
     Main B is reading zip (zero) right now."
  ],
  topics: [ "breakdown", "apollo" ],
  keywords: { "make": "saturn", "type": "rocket" }
}
```

Categories / Tags / Topics / Keywords

```sql
CREATE TABLE call (
  id            UUID      NOT NULL,
  phone         TEXT      NOT NULL,
  transcript    JSON      NOT NULL,
  ...
  topics        TEXT[]            ,
);
```

Categories / Tags / Topics / Keywords

```
SELECT *
FROM call
WHERE topics @> ARRAY['breakdown'];

SELECT *
FROM call
WHERE topics @> ARRAY['breakdown', 'apollo'];
```

Categories / Tags / Topics / Keywords

```
CREATE TABLE comms.call (
  id              UUID       NOT NULL,
  phone           TEXT       NOT NULL,
  transcript      JSON       NOT NULL,
  ...
  keywords        JSONB              ,
);
```

Categories / Tags / Topics / Keywords

```
SELECT *
FROM comms.call
WHERE keywords @>
         '{"make": "saturn"}'::JSONB;
```

Categories / Tags / Topics / Keywords

```
CREATE INDEX topics_idx
ON call USING GIN (topics);

CREATE INDEX keywords_idx
ON call USING GIN (keywords);
```

Modern SQL

Simple Analytics

```
CREATE TABLE quote (
  id          UUID        NOT NULL,
  customer_id UUID        NOT NULL,
  status      STATUS      NOT NULL,
  price       NUMERIC     NOT NULL,
  answers     JSONB
);
```

Simple Analytics

```
SELECT count(*),
       count(*) FILTER (WHERE (answers ->> 'locks')
                           IS NULL),
       count(*) FILTER (WHERE (answers ->> 'locks')
                           IS NOT NULL),
       count(*) FILTER (WHERE (answers ->> 'locks')
                           = '3-lever'),
       count(*) FILTER (WHERE (answers ->> 'locks')
                           = 'unknown')
FROM insurance.quotes;
```

Bucketing Data

```
CREATE TABLE device_readings (
  device_id     UUID       NOT NULL,
  time          TIMESTAMP  NOT NULL,
  temperature   NUMERIC            ,
  light         NUMERIC            ,
  PRIMARY KEY (meter_id, day)
);
```

Bucketing Data

```
SELECT r.device_id, t.time, array_agg(r.read_at),
       avg(r.temperature), avg(r.light)
FROM generate_series(
   '2022-10-06 00:00:00'::TIMESTAMP,
   '2022-10-07 00:00:00'::TIMESTAMP, '10 minutes') t(time)
JOIN device_readings r
   ON (r.device_id = '26170b53-ae8f-464e-8ca6-2faeff8a4d01'::UUID
       AND r.read_at >= t.time
       AND r.read_at < (t.time + '10 minutes'))
GROUP BY 1, 2
ORDER BY t.time;
```

Energy Meter

```
CREATE TABLE daily_reading (
  meter_id          UUID        NOT NULL,
  day               DATE        NOT NULL,
  energy            BIGINT              ,
  area              BIGINT              ,
  PRIMARY KEY (meter_id, day)
);
```

Window Functions - Counters

```sql
SELECT
 day,
 energy,
 energy - coalesce(lag(energy)
    OVER (ORDER BY day), 0) AS consumed
FROM daily_reading
ORDER BY day;
```

Window Functions - Roll Up

```sql
WITH daily_consumption AS (...)
SELECT consumed AS daily_consumed,
       sum(consumed) OVER
       (PARTITION BY
         date_trunc('week', day))
       AS weekly_consumed
FROM daily_consumption;
```

Window Functions - Moving On Up

```
WITH daily_consumption AS (...)
SELECT consumed AS daily_consumed,
       avg(consumed) OVER
       (ORDER BY day
         ROWS BETWEEN 2 PRECEDING
         AND CURRENT ROW)
       AS moving_average
FROM daily_consumption;
```

Recursion - Everything In An Area

```
CREATE TABLE area (
  id          BIGINT                ,
  name        TEXT                  ,
  parent_id   BIGINT                ,
  PRIMARY KEY (id)
);
```

Recursion - Everything In An Area

```
WITH RECURSIVE areas(id) AS (
      SELECT a.id FROM area a
      WHERE a.name = ‘West Midlands’
    UNION
      SELECT a.id FROM area a, areas aa
      WHERE a.parent_id = aa.id
)
SELECT r.*
FROM areas a
JOIN daily_reading r ON (r.area = a.id)
```

Writable CTEs

```
CREATE TABLE commission_record (
  customer_id  UUID         NOT NULL,
  logged_at    TIMSTAMPTZ   NOT NULL,
  value        NUMERIC      NOT NULL,
  invoice_id   BIGINT
);
```

Writable CTEs

```
WITH invoice_commission AS (
    UPDATE commission_record
    SET invoice_id = 123
    WHERE invoice_id IS NULL
      AND customer_id = $1
    RETURNING *
)
INSERT INTO invoice
SELECT 123, current_date, sum(value) AS total
FROM invoice_commission;
```

Writable CTEs

```sql
SELECT t.*, q.*
FROM tenant t
LEFT JOIN LATERAL (
    SELECT invoice_date, total
    FROM invoice i
    WHERE i.tenant_id = t.id
    ORDER BY invoice_date DESC
    LIMIT 1
) q ON (true);
```

Staying In Control

There Can Only Be One

```
CREATE TABLE subscription (
  id          UUID      NOT NULL,
  member_id   UUID      NOT NULL,
  plan_id     UUID      NOT NULL,
  status      STATUS    NOT NULL,
  ...
);
```

There Can Only Be One

```
CREATE UNIQUE INDEX active_subs
ON club.subscription
  (member_id)
WHERE status = 'active';
```

Controlling JSON Data

```
CREATE TABLE quote (
  id          UUID        NOT NULL,
  customer_id UUID        NOT NULL,
  status      STATUS      NOT NULL,
  price       NUMERIC     NOT NULL,
  answers     JSONB
);
```

Controlling JSON Data

```
ALTER TABLE insurance.quote
ADD CONSTRAINT answers_chk
CHECK (
  jsonb_typeof( answers ) = 'object'
);
```

Scheduled Tasks

Tasks - Simple Scheduled Tasks

```sql
CREATE TABLE task (
  id          UUID        PRIMARY KEY,
  after       TIMESTAMP   NOT NULL,
  processed   BOOLEAN     NOT NULL,
  attempt     INTEGER     NOT NULL,
  task_name   TEXT        NOT NULL,
  value       JSONB       NOT NULL
);
```

Tasks - Execute

```
SELECT *
FROM task
WHERE NOT processed AND after < now()
ORDER BY after DESC
LIMIT 1
FOR UPDATE SKIP LOCKED;
```

Tasks - Partial Indexes

```
CREATE INDEX task_after_idx
ON task (after)
WHERE NOT processed;
```

Tasks - Partial Indexes

```
Limit
    (cost=0.14..2.98 rows=1 width=463)
    (actual time=0.054..0.055 rows=0 loops=1)
  ->  LockRows
          (cost=0.14..45.69 rows=16 width=463)
          (actual time=0.052..0.052 rows=0 loops=1)
        ->  Index Scan Backward using task_after_idx on task
                (cost=0.14..45.53 rows=16 width=463)
                (actual time=0.050..0.051 rows=0 loops=1)
              Index Cond: (after < now())
              Filter: (NOT processed)
Planning Time: 0.276 ms
Execution Time: 0.115 ms
```

Task - Retry

```sql
UPDATE task
SET after = now() + '1 hour'::INTERVAL,
    attempt = attempt + 1
WHERE id = '...';
```

Tasks - Processed / Failed

```
UPDATE task
SET processed = true
WHERE id = '...';
```

Fuzzy Matching

```
Code,        Category, Title,                  Description
LVB412-255, DOOR,     DOOR FRAME - DENTED,    ...
LVB412-591, DOOR,     DOOR - WILL NOT CLOSE, ...
LVB412-259, DOOR,     DOOR OPENS MID-CYCLE,   ...
```

```sql
CREATE TABLE fault_code (
  id            UUID        NOT NULL,
  category      TEXT        NOT NULL,
  title         TEXT        NOT NULL,
  description   TEXT                ,
);
```

Text Search - Simple

```
SELECT *
FROM fault_code
WHERE
 to_tsvector('english',
   title || ' ' || coalesce(description, '')
 )
 @@ to_tsquery('english', 'leak');
```

Text Search - Simple Yet Fast

```
CREATE INDEX fc_text_idx
ON fault_code
USING GIN
(to_tsvector('english',
   title || ' ' || coalesce(description, '')
));
```

Text Search - Simple Yet Fast

```
Seq Scan on fault_code  (cost=0.00..870.51 rows=15
width=170) (actual time=0.084..24.966 rows=37
loops=1)
  Rows Removed by Filter: 2978
Planning Time: 0.172 ms
Execution Time: 25.069 ms
```

Text Search - Simple Yet Fast

```
Bitmap Heap Scan on fault_code  (cost=3.03..22.53
rows=15 width=170) (actual time=0.044..0.167 rows=37
loops=1)
  Heap Blocks: exact=20
  ->  Bitmap Index Scan on fc_text_idx
(cost=0.00..3.03 rows=15 width=0) (actual
time=0.027..0.028 rows=37 loops=1)
Planning Time: 0.308 ms
Execution Time: 0.271 ms
```

Text Search - Realistic

```
ALTER TABLE fault_code
 ADD COLUMN vector TSVECTOR;

CREATE INDEX fc_vector_idx
ON fault_code
USING GIN (vector);
```

Text Search - Realistic

```
UPDATE reference.fault_code
SET vector =
   setweight(
     to_tsvector(coalesce(title,'')), 'A'
   ) ||
   setweight(
     to_tsvector(coalesce(description,'')), 'B'
   );
```

Text Search - Realistic

```
SELECT
  ts_rank_cd(vector,
    websearch_to_tsquery(…)), *
FROM fault_code
WHERE vector @@ websearch_to_tsquery(
  'english', 'leaking door')
ORDER BY 1;
```

Fuzzy Matching Names

```
CREATE EXTENSION fuzzystrmatch;

SELECT soundex('Matthew'), soundex('Matt');
---------+----------
 M300    | M300

SELECT dmetaphone('John'), dmetaphone('Jon');
------------+-------------
 JN         | JN
```

Fuzzy Matching Names

```
CREATE INDEX user_first_dm
  ON users (dmetaphone(first_name));

SELECT * FROM users
WHERE dmetaphone(first_name)
        = dmetaphone('Jon');
------------+
 Jon        |
 John       |
```

Fuzzy Matching Names

```
CREATE INDEX user_fn_dmt ON users
  USING GIN (daitch_mokotoff(first_name));

SELECT * FROM users
WHERE daitch_mokotoff(first_name)
        && daitch_mokotoff('Chris');
------------+
 Chris      |
 Kris       |
```

GIS

Location Search

```
CREATE TABLE club.venue (
  id          UUID       NOT NULL,
  name        TEXT       NOT NULL,
  description TEXT       NOT NULL,
  address     TEXT       NOT NULL,
  location    Geometry(POINT, 4326)
);
```

Location Search

```sql
SELECT *
FROM venue
WHERE st_dwithin(location, $1, 2000);
```

Location Matching

```
CREATE TABLE engineer (
   id     UUID                  NOT NULL,
   name   TEXT                  NOT NULL,
   area   Geometry(MultiPolygon, 4326)
);
```

Location Matching

```sql
SELECT *
FROM engineer
WHERE st_contains(area, $1);
```

Location Matching

```sql
SELECT *
FROM engineer
WHERE st_intersects(area,
    st_buffer(
        st_point(-71.104, 42.315, 4326),
        0.025
    )
);
```

Location Search / Matching - Faster

```
CREATE INDEX venue_location_idx
ON venue GIST (location);
```

All Together Now

All Together Now

```
CREATE TABLE search.content (
  id         UUID,
  vector     TSVECTOR,
  tags       TEXT[],
  location   Geometry(POINT, 4326)
);
```

All Together Now

```
SELECT *
FROM content
WHERE vector @@ to_tsquery('library')
AND st_dwithin(location, my_location, 2000)
AND tags @> ARRAY['service_catalogue'];
```

It doesn't have to be like this; All we need to do is make sure we keep talking

Mind The Gap

Mind The Gap

```
CREATE TABLE meter_reading (
  meter_id          BIGINT  NOT NULL,
  day               DATE    NOT NULL,
  energy            BIGINT          ,
  PRIMARY KEY (meter_id, day)
);
```

Mind The Gap

```
WITH days AS (
  SELECT t.day::DATE
  FROM generate_series('2017-01-01'::DATE,
'2017-01-15'::DATE, '1 day') t(day)
), data AS (
   SELECT *
   FROM meter_reading
   WHERE meter_id = 123
   AND day >= '2017-01-01'::DATE
   AND   day <= '2017-01-15'::DATE
)
```

Mind The Gap

```
SELECT day,
       coalesce(energy,
         (((next_read - last_read)
            / (next_read_time - last_read_time))
            * (day - last_read_time))
            + last_read) AS energy_interpolated
FROM (
    … from next slide …
) q
ORDER BY day
```

Mind The Gap

```
SELECT t.day, d.energy,
  last(d.day)    OVER lookback    AS last_read_time,
  last(d.day)    OVER lookforward AS next_read_time,
  last(d.energy) OVER lookback    AS last_read,
  last(d.energy) OVER lookforward AS next_read
FROM days t
LEFT JOIN data d ON (t.day = d.day)
WINDOW
  lookback AS (ORDER BY t.day),
  lookforward AS (ORDER BY t.day DESC)
```

Mind The Gap

```
CREATE FUNCTION last_agg(anyelement, anyelement)
RETURNS anyelement LANGUAGE SQL IMMUTABLE STRICT AS $$
      SELECT $2;
$$;

CREATE AGGREGATE last (
      sfunc = last_agg,
      basetype = anyelement,
      stype = anyelement
);
```