

Postgres starter kit (PDF)

A worked Postgres schema from three SharePoint lists, three queries across lookups, a Metabase connection and row-level security for an AI agent's credential.

A facilities team runs its work in three SharePoint lists. Buildings holds each site and its region, Contractors holds suppliers and their contract end dates, and Work Orders holds the jobs, with a lookup to each of the other two. The questions they keep asking span all three lists, and no single list view can answer them. Every example below comes from that setup, and the SQL runs on any Postgres, including Supabase and Azure Database for PostgreSQL.

The schema SPSync creates

With a site prefix of fm and a destination schema called spsync, the three lists become these tables. Column names come from SharePoint's internal names, lower-cased.

Table | Key columns

fm_buildings | sp_item_id, title, region

fm_contractors | sp_item_id, title, contract_end

fm_work_orders | sp_item_id, title, status, cost, due_date, building_id, building_value, contractor_id, contractor_value, assigned_to_user_id

fm_work_orders__trades | sp_item_id, sp_site_id, value, ordinal

sp_users | id, login, display_name, email

Each lookup gives two columns, the target's item id and its display text. Trades is a multi-select choice, so it gets its own child table with one row per trade per job. Every list table also carries sp_created, sp_modified, sp_editor_id and sp_deleted_at. A deleted item keeps its row with sp_deleted_at set, so filter on sp_deleted_at is null, or turn on reporting views and query v_fm_work_orders, which leaves deleted rows out and adds the assignee's name.

Three queries across lookups

Open and overdue jobs per building

Status is a single choice column, so the filter is a plain string comparison. Overdue means a due date before today.

```
select b.title as building,
       count(*) as open_jobs,
       count(*) filter (where w.due_date < current_date) as overdue
from spsync.fm_work_orders w
join spsync.fm_buildings b on b.sp_item_id = w.building_id
where w.sp_deleted_at is null and w.status <> 'Closed'
```

```
group by b.title
order by overdue desc;
```

Spend against contracts that end soon

This one totals the quarter's spend for each contractor whose contract ends within 90 days. Cost is a currency column, which arrives as numeric(38,9), so the sum stays exact.

```
select c.title as contractor, c.contract_end, sum(w.cost) as spend
from spsync.fm_work_orders w
join spsync.fm_contractors c on c.sp_item_id = w.contractor_id
where w.sp_deleted_at is null
      and w.due_date >= date_trunc('quarter', current_date)
      and c.contract_end < current_date + 90
group by c.title, c.contract_end
order by c.contract_end;
```

Electrical jobs per engineer

The trade filter goes through the child table and the name comes from sp_users. A job tagged with two trades appears once per trade in the child table.

```
select u.display_name, count(*) as electrical_jobs
from spsync.fm_work_orders w
join spsync.fm_work_orders__trades t on t.sp_item_id = w.sp_item_id
join spsync.sp_users u on u.id = w.assigned_to_user_id
where w.sp_deleted_at is null and t.value = 'Electrical'
group by u.display_name;
```

The second query is the one people tend to rebuild in Excel every month. In SharePoint it needs two exports and a VLOOKUP, because a Work Orders view can't filter on a date that lives in Contractors.

Connecting Metabase or Grafana

Give each tool its own login with read access only. For Metabase that takes three statements.

```
create role metabase_reader login password 'change-me';
grant usage on schema spsync to metabase_reader;
grant select on all tables in schema spsync to metabase_reader;
```

In Metabase, add a PostgreSQL database under Admin settings, enter the host, port, database name and the metabase_reader login, turn on SSL, and limit the schemas to spsync. The three queries above go in as native SQL questions and sit together on one dashboard. Grafana works the same way through its PostgreSQL data source, with TLS set to require for a hosted database. A new list added to the replication later gives you a new table, so run the grant select line again after that or the dashboard won't see it.

Row-level security for an agent credential

An AI agent should get its own login, read only, with the rows it may see decided by the database.

SPSync doesn't copy SharePoint's item permissions, so this is where you set the boundary. Say the agent serves the North region only:

```
create role agent_north login password 'change-me';
alter role agent_north set default_transaction_read_only = on;
alter role agent_north set statement_timeout = '15s';
grant usage on schema spsync to agent_north;
grant select on spsync.fm_work_orders, spsync.fm_buildings to agent_north;
alter table spsync.fm_work_orders enable row level security;
create policy north_only on spsync.fm_work_orders for select to agent_north
    using (building_id in (select sp_item_id from spsync.fm_buildings where region =
        'North'));
create policy reporting_all on spsync.fm_work_orders for select to metabase_reader
    using (true);
```

Two details catch people out. Once row-level security is on, a role with no policy sees no rows at all, which is why the reporting_all policy is there for Metabase. The sync writes as the table owner, and owners skip these policies, so leave force row level security off or the sync's own updates will be filtered.

To check the boundary, connect as agent_north and run the first query above. Only North buildings should appear, and any update should fail.

Which Postgres MCP server should the agent use?

To give Claude, Cursor or another assistant SQL tools over this replica, point a Postgres MCP server at the database with the agent_north login (SPSync's own read-only spsync mcp serve takes the same kind of login), and the policy applies to every query it runs. Turn on the server's own read-only setting as well, as a second layer.

DBHub, from Bytebase, takes readonly = true and a max_rows cap on its execute_sql tool in dbhub.toml, and also serves SQL Server, MySQL and SQLite if the replica moves:

```
[[sources]]
id = "facilities"
dsn =
"postgres://agent_north:${AGENT_NORTH_PASSWORD}@db.internal:5432/facilities?sslmode=require"
[[tools]]
name = "execute_sql"
source = "facilities"
readonly = true
max_rows = 200
```

Claude Desktop or Cursor then starts it with the full path to that file:

```
{
  "mcpServers": {
    "facilities-replica": {
      "command": "npx",
      "args": ["-y", "@bytebase/dbhub@1.5.0", "--config", "/path/to/dbhub.toml"],
      "env": {
```

```
    "AGENT_NORTH_PASSWORD": "<password>"
  }
}
}
```

With version 1.5.0 and readonly = true, an update comes back as READONLY_VIOLATION before it reaches the database. Earlier copies of this kit used Postgres MCP Pro from Crystal DBA. On 23 October 2026 its current release, 0.3.0, would not install, because it requires pglast 7.2.0 and PyPI no longer has that version, so uvx postgres-mcp stops with a resolver error. The Model Context Protocol project's original reference Postgres server is archived; skip it for anything beyond a tutorial. Whichever server you use, the role and the policy above are the boundary, because they hold even if the server's own check is bypassed.

The SharePoint to PostgreSQL walkthrough shows the setup that produces these tables, step by step, and the MCP post works through a full assistant session over a replica like this one.

Full, up to date version: <https://spsync.io/resources/postgres-starter-kit/>