Hasura Training

Pre-Work

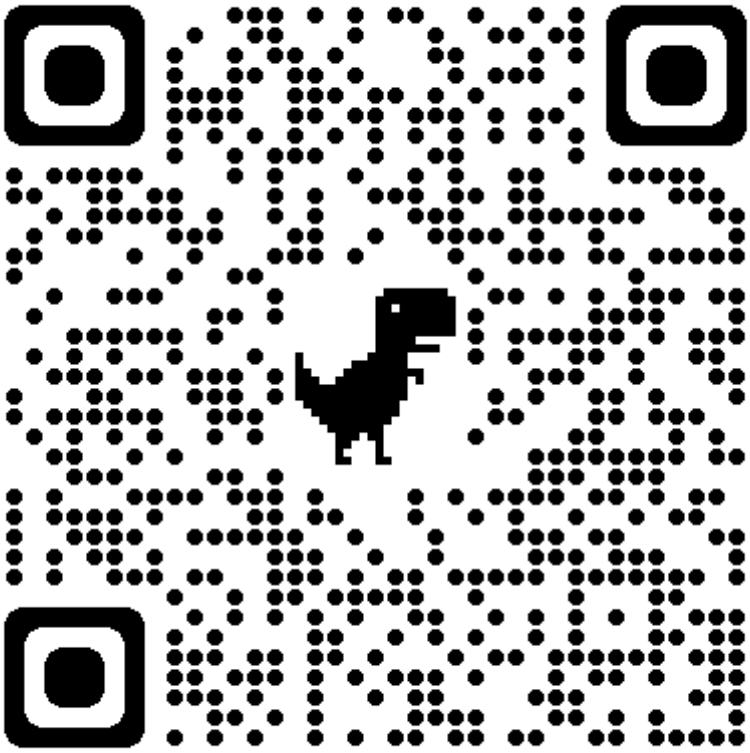
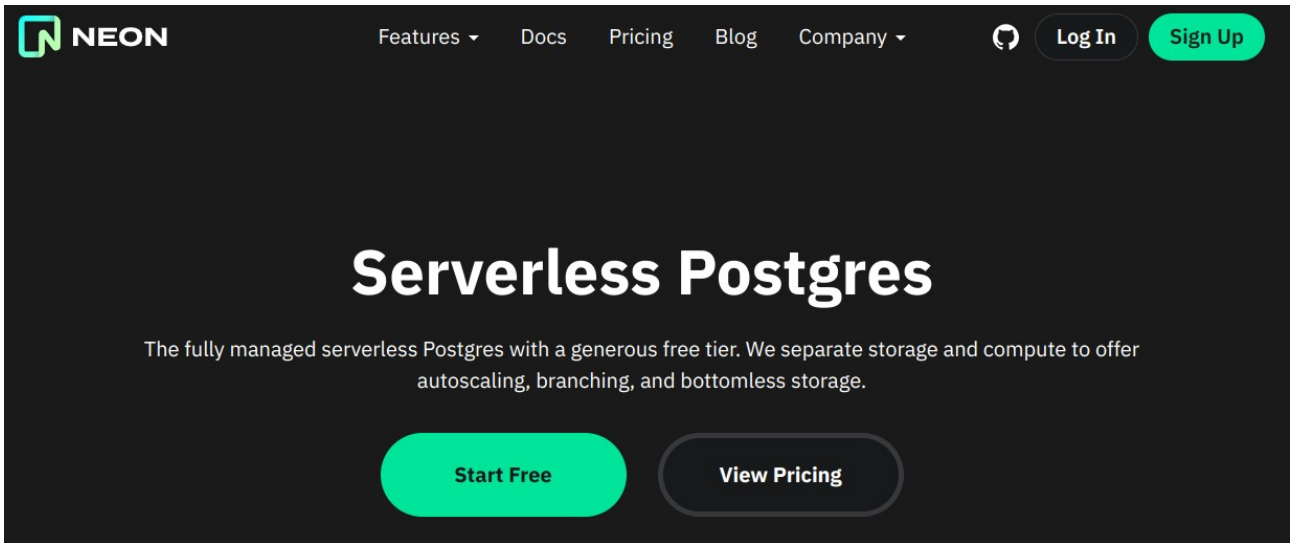
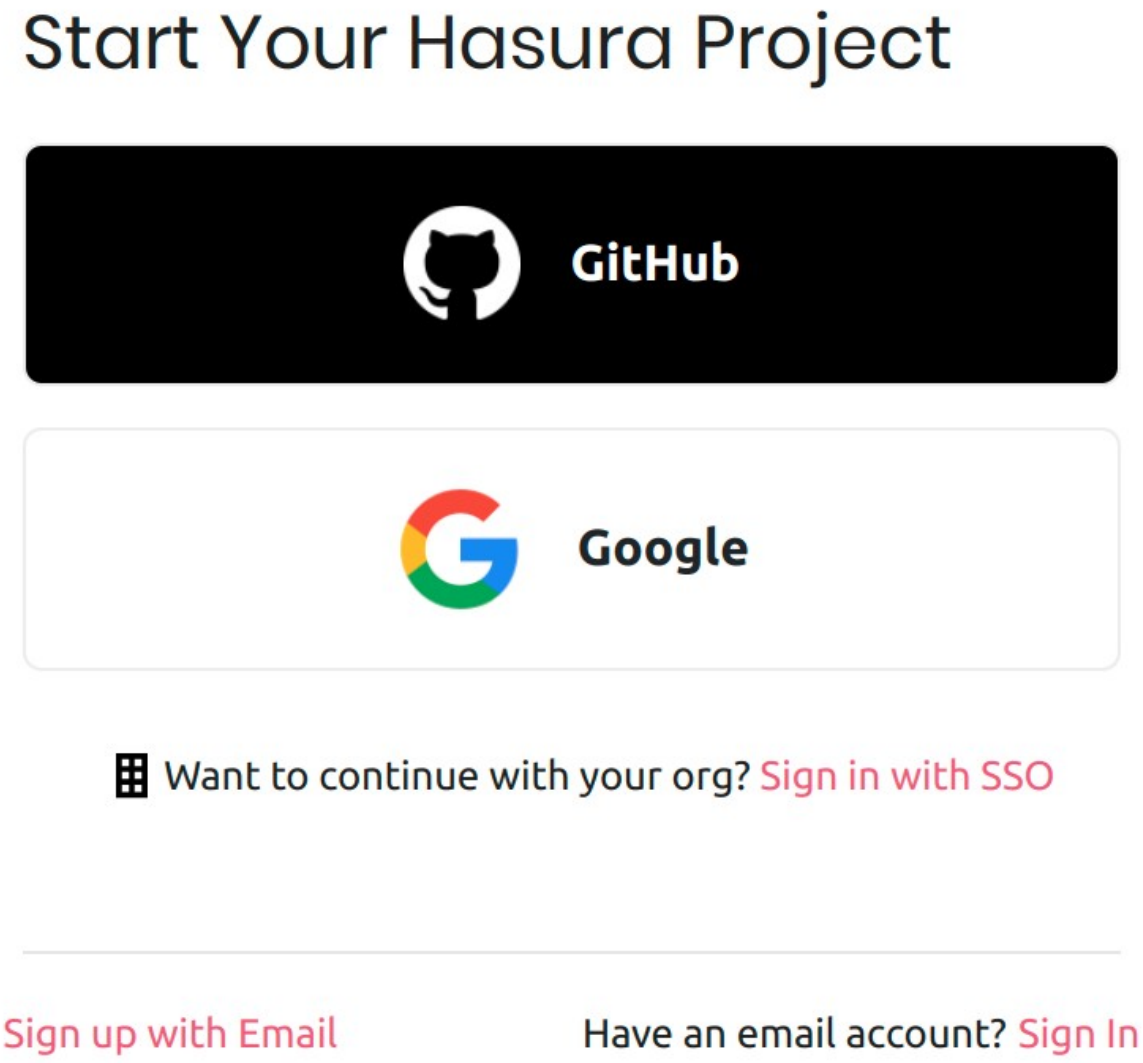


Figure 1: https://tinyurl.com/ytxjjck7

Step 1: Log into Neon

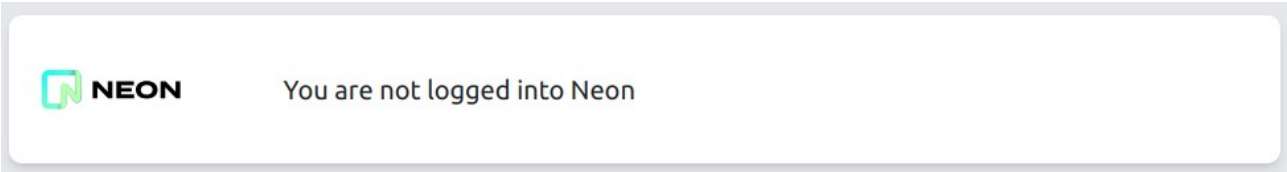


Step 2: Log into Hasura Cloud



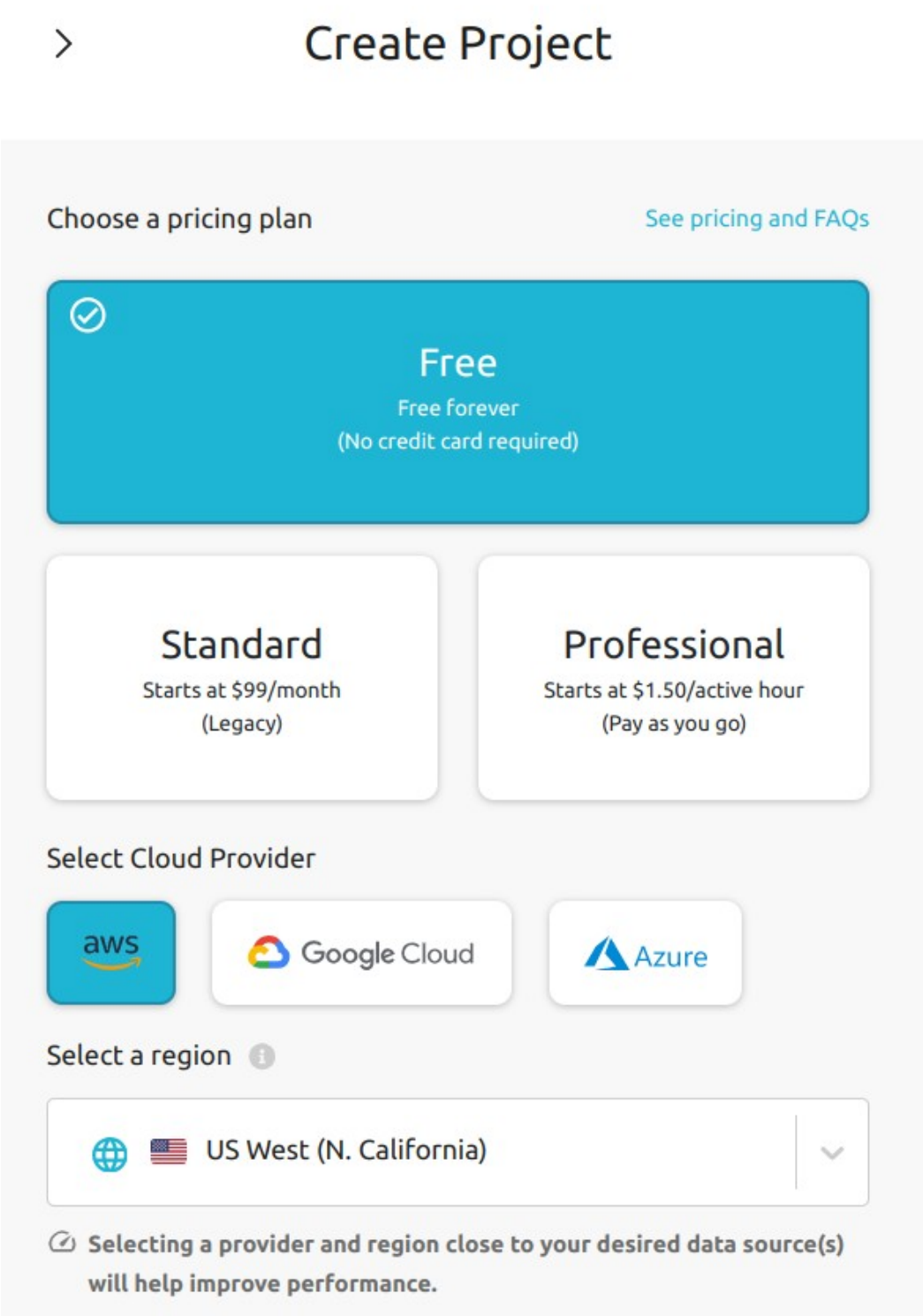
Step 3: Connect Neon to Hasura Cloud

My Account › Connections



Step 4: Create a new Hasura Cloud Project

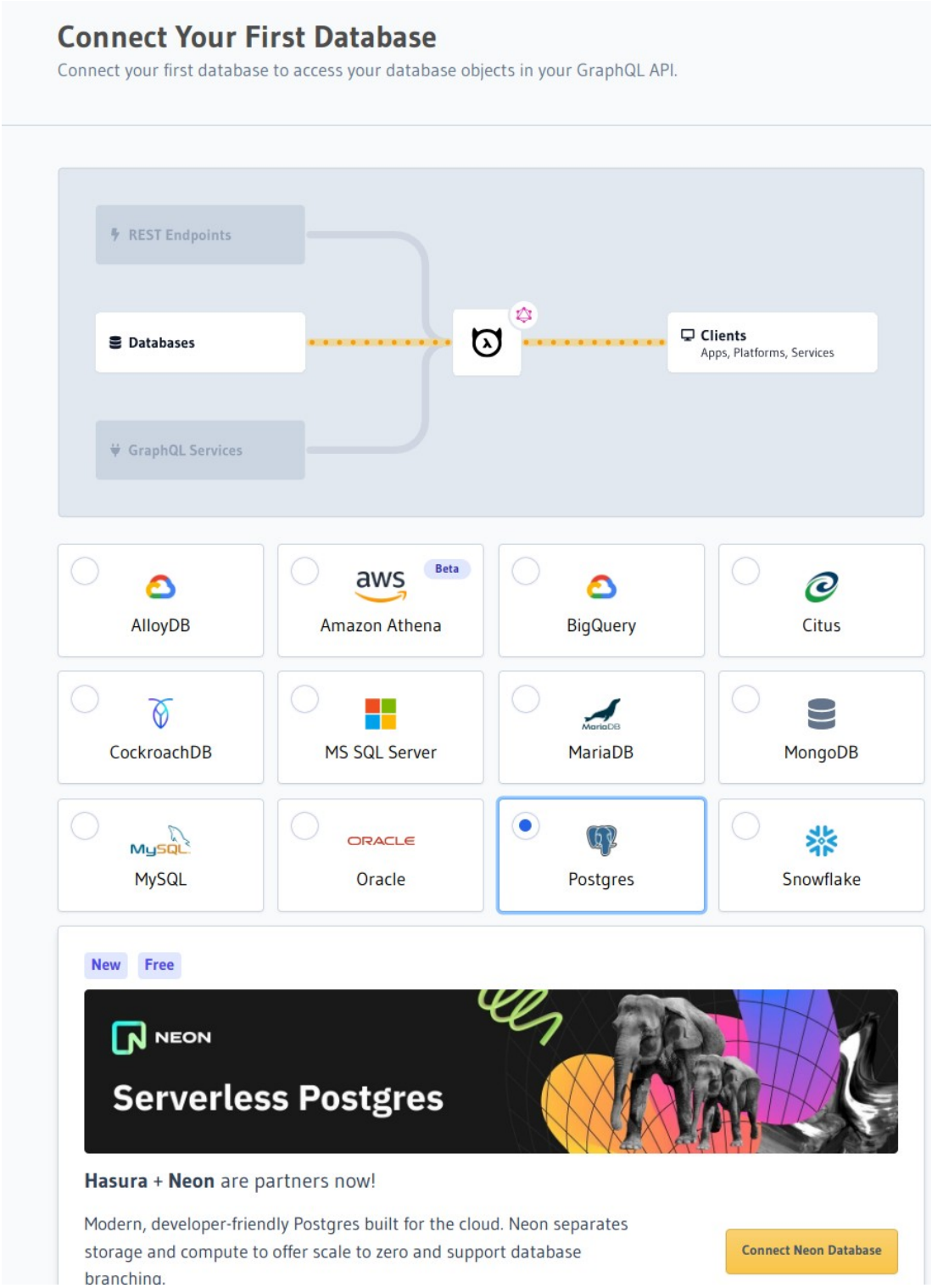
Projects › New Project



Step 5: Launch Console

Step 6: Create a new Neon database

Console › Data › Connect Your First Database



Step 7: Create the Data Model

Console › Data › Run SQL

https://tinyurl.com/367p55ca

```
-- -*- sql-product: postgres; -*-
```

```sql
CREATE EXTENSION IF NOT EXISTS pgcrypto;

-- account table

CREATE TABLE "public"."account" ("id" uuid NOT NULL DEFAULT gen_random_uuid(),
"name" text NOT NULL, "created_at" timestamptz NOT NULL DEFAULT now(),
"updated_at" timestamptz NOT NULL DEFAULT now(), PRIMARY KEY ("id") );
CREATE OR REPLACE FUNCTION "public"."set_current_timestamp_updated_at"()
  RETURNS TRIGGER AS $$
  DECLARE
    _new record;
  BEGIN
    _new := NEW;
    _new."updated_at" = NOW();
    RETURN _new;
  END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER "set_public_account_updated_at"
  BEFORE UPDATE ON "public"."account"
  FOR EACH ROW
  EXECUTE PROCEDURE "public"."set_current_timestamp_updated_at"();
COMMENT ON TRIGGER "set_public_account_updated_at" ON "public"."account"
  IS 'trigger to set value of column "updated_at" to current timestamp on row
update';

-- product table

CREATE TABLE "public"."product" ("id" uuid NOT NULL DEFAULT gen_random_uuid(),
"created_at" timestamptz NOT NULL DEFAULT now(), "updated_at" timestamptz NOT
NULL DEFAULT now(), "name" text NOT NULL, "price" integer NOT NULL, PRIMARY KEY
("id") );
CREATE OR REPLACE FUNCTION "public"."set_current_timestamp_updated_at"()
  RETURNS TRIGGER AS $$
  DECLARE
    _new record;
  BEGIN
    _new := NEW;
    _new."updated_at" = NOW();
    RETURN _new;
  END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER "set_public_product_updated_at"
  BEFORE UPDATE ON "public"."product"
  FOR EACH ROW
  EXECUTE PROCEDURE "public"."set_current_timestamp_updated_at"();
COMMENT ON TRIGGER "set_public_product_updated_at" ON "public"."product"
  IS 'trigger to set value of column "updated_at" to current timestamp on row
update';

-- order table

CREATE TABLE "public"."order" ("id" uuid NOT NULL DEFAULT gen_random_uuid(),
"created_at" timestamptz NOT NULL DEFAULT now(), "updated_at" timestamptz NOT
NULL DEFAULT now(), "account_id" uuid NOT NULL, PRIMARY KEY ("id") , FOREIGN
KEY ("account_id") REFERENCES "public"."account"("id") ON UPDATE restrict ON
DELETE restrict);
CREATE OR REPLACE FUNCTION "public"."set_current_timestamp_updated_at"()
  RETURNS TRIGGER AS $$
  DECLARE
    _new record;
  BEGIN
```

```
    _new := NEW;
    _new."updated_at" = NOW();
    RETURN _new;
  END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER "set_public_order_updated_at"
  BEFORE UPDATE ON "public"."order"
  FOR EACH ROW
  EXECUTE PROCEDURE "public"."set_current_timestamp_updated_at"();
COMMENT ON TRIGGER "set_public_order_updated_at" ON "public"."order"
  IS 'trigger to set value of column "updated_at" to current timestamp on row
update';

create index on "order" (account_id);

-- order_detail table

CREATE TABLE "public"."order_detail" ("id" uuid NOT NULL DEFAULT
gen_random_uuid(), "created_at" timestamptz NOT NULL DEFAULT now(),
"updated_at" timestamptz NOT NULL DEFAULT now(), "units" integer NOT NULL,
"order_id" uuid NOT NULL, "product_id" uuid NOT NULL, PRIMARY KEY ("id") ,
FOREIGN KEY ("order_id") REFERENCES "public"."order"("id") ON UPDATE restrict
ON DELETE restrict, FOREIGN KEY ("product_id") REFERENCES
"public"."product"("id") ON UPDATE restrict ON DELETE restrict);
CREATE OR REPLACE FUNCTION "public"."set_current_timestamp_updated_at"()
  RETURNS TRIGGER AS $$
  DECLARE
    _new record;
  BEGIN
    _new := NEW;
    _new."updated_at" = NOW();
    RETURN _new;
  END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER "set_public_order_detail_updated_at"
  BEFORE UPDATE ON "public"."order_detail"
  FOR EACH ROW
  EXECUTE PROCEDURE "public"."set_current_timestamp_updated_at"();
COMMENT ON TRIGGER "set_public_order_detail_updated_at" ON
"public"."order_detail"
  IS 'trigger to set value of column "updated_at" to current timestamp on row
update';

create index on order_detail (order_id);

create index on order_detail (product_id);

-- product_search function

create or replace function product_search(search text)
  returns setof product as $$
  select product.*
  from product
  where
  name ilike ('%' || search || '%')
$$ language sql stable;

-- product_search_slow function

create or replace function product_search_slow(search text, wait real)
  returns setof product as $$
```

```
  select product.*
  from product, pg_sleep(wait)
  where
  name ilike ('%' || search || '%')
$$ language sql stable;

-- non_negative_price constraint

alter table "public"."product" add constraint "non_negative_price" check (price
> 0);

-- index account(name)

create index if not exists account_name_idx on account (name);

-- status enum

CREATE TYPE status AS ENUM ('new', 'processing', 'fulfilled');

-- add status to order table

alter table "public"."order" add column "status" status null;

create index on "order" (status);

-- region dictionary table

create table if not exists region (
  value text primary key,
  description text);

-- add region to order

alter table "public"."order" add column "region" Text
 null;

alter table "public"."order"
  add constraint "order_region_fkey"
  foreign key ("region")
  references "public"."region"
  ("value") on update restrict on delete restrict;

create index on "order" (region);
```

Step 8: Insert Sample Data

Console › Data › Run SQL

https://tinyurl.com/392tvc9p

```
-- -*- sql-product: postgres; -*-
insert into account (name) values ('Christel Seaborn');
insert into account (name) values ('Emalia Oliveras');
insert into account (name) values ('Arin Maker');
insert into account (name) values ('Gregor Gwilliam');
insert into account (name) values ('Calypso Meyer');

insert into product (name, price) values ('Plastic Wrap', 71);
insert into product (name, price) values ('Arizona - Green Tea', 377);
insert into product (name, price) values ('Wine - Prosecco Valdobienne', 220);
```

```
insert into product (name, price) values ('Sproutsmustard Cress', 771);
insert into product (name, price) values ('Spinach - Baby', 740);

with
  account as (
    select
      account.id,
      name,
      (random()*5)::int orders
      from account)
    insert into "order" (account_id)
select
  account_id
  from (
    select
      account.id account_id,
      row_number() over (partition by account.id order by account.name) ordinal
      from account, generate_series(1, 5)) orders
       join account on account.id = orders.account_id
           and orders.ordinal <= account.orders;

with
  "order" as (
    select
      "order".id,
      (random()*9 + 1)::int items
      from "order")
    insert into order_detail (order_id, product_id, units)
select
  order_id,
  product_id,
  (random()*9 + 1)::int units
  from (
    select
      "order".id order_id,
      product.id product_id,
      row_number() over (partition by "order".id) ordinal
      from "order", product) user_item
       join "order" on "order".id = user_item.order_id
           and user_item.ordinal <= "order".items;

update "order" set status = ((array['new', 'processing', 'fulfilled'])
[floor(random()*3+1)])::status;

insert into region (value, description)
values
  ('NORTHEAST', 'New England'),
  ('MIDWEST', 'Great Lakes'),
  ('SOUTH', 'Dixie'),
  ('PLAINS', 'Great Plains'),
  ('APPALACHIA', 'Pennsylvania and West Virginia'),
  ('MOUNTAIN', 'Rocky Mountains'),
  ('NORTHWEST', 'Rainy'),
  ('WEST', 'California'),
  ('SOUTHWEST', 'Cacti');

update "order" set region = ((array[
  'NORTHEAST',
  'MIDWEST',
  'SOUTH',
  'PLAINS',
```

```
  'APPALACHIA',
  'MOUNTAIN',
  'NORTHWEST',
  'WEST',
  'SOUTHWEST'
  ])[floor(random()*9+1)])::text;

commit;
```