-- Demonstrates race conditions

-- Connection 1
SELECT "balance" FROM "accounts" WHERE "id" = 3;

-- Connection 2
SELECT "balance" FROM "accounts" WHERE "id" = 3;

-- Connection 1
UPDATE "accounts" SET "balance" = "balance" + 30 WHERE "id" = 1;

-- Connection 2
UPDATE "accounts" SET "balance" = "balance" + 30 WHERE "id" = 1;

-- Connection 3
SELECT "balance" FROM "accounts" WHERE "id" = 1;
UPDATE "accounts" SET "balance" = "balance" - 30 WHERE "id" = 1;

-- Connection 1
UPDATE "accounts" SET "balance" = "balance" - 30 WHERE "id" = 3;

-- Connection 2
UPDATE "accounts" SET "balance" = "balance" - 30 WHERE "id" = 3;
-- Demonstrates atomicity of transactions
-- bank.db

-- Show schema, highlight CHECK constraint
.schema

-- View account balances
SELECT * FROM "accounts";

-- Update balance without a transaction
UPDATE "accounts" SET "balance" = "balance" + 10 WHERE "id" = 2;
SELECT * FROM "accounts"; -- Viewing here provides an improper view of total balances
UPDATE "accounts" SET "balance" = "balance" - 10 WHERE "id" = 1;
SELECT * FROM "accounts"; -- Viewing here, after all updated, results in proper view

-- Reset table
UPDATE "accounts" SET "balance" = "balance" - 10 WHERE "id" = 2;
UPDATE "accounts" SET "balance" = "balance" + 10 WHERE "id" = 1;

-- Create a transaction which is successful
-- View state of database from other terminal mid-way through transaction
BEGIN TRANSACTION;
UPDATE "accounts" SET "balance" = "balance" + 10 WHERE "id" = 2;
UPDATE "accounts" SET "balance" = "balance" - 10 WHERE "id" = 1;
COMMIT;

-- Complete invalid update of balance without a transaction
UPDATE "accounts" SET "balance" = "balance" + 10 WHERE "id" = 2;
UPDATE "accounts" SET "balance" = "balance" - 10 WHERE "id" = 1; -- Invokes constraint error, which is ABORTed

-- "Rollback" the balance
UPDATE "accounts" SET "balance" = "balance" - 10 WHERE "id" = 2;

-- Create a transaction which should be rolled back
BEGIN TRANSACTION;
UPDATE "accounts" SET "balance" = "balance" + 10 WHERE "id" = 2;
UPDATE "accounts" SET "balance" = "balance" - 10 WHERE "id" = 1; -- Invokes constraint error, which is ABORTed
ROLLBACK;
-- Demonstrates foreign key indexes

-- movies.db

-- Time searching for movies Tom Hanks has starred in
.timer on
SELECT "title" FROM "movies" WHERE "id" IN (  SELECT "movie_id" FROM "stars" WHERE "person_id" = (    SELECT "id" FROM "people" WHERE "name" = 'Tom Hanks'  )  );
.timer off

-- Identify which columns we should create indexes on
EXPLAIN QUERY PLAN
SELECT "title" FROM "movies" WHERE "id" IN (  SELECT "movie_id" FROM "stars" WHERE "person_id" = (    SELECT "id" FROM "people" WHERE "name" = 'Tom Hanks'  )  );

-- Create index on foreign key
.timer on
CREATE INDEX "person_index" ON "stars" ("person_id");
.timer off

-- Create index to speed name look-ups
CREATE INDEX "name_index" ON "people" ("name");
.timer off

EXPLAIN QUERY PLAN
SELECT "title" FROM "movies" WHERE "id" IN (  SELECT "movie_id" FROM "stars" WHERE "person_id" = (    SELECT "id" FROM "people" WHERE "name" = 'Tom Hanks'  )  );

-- Make person_index a covering index for the above query
CREATE INDEX "person_index" ON "stars" ("person_id", "movie_id");

EXPLAIN QUERY PLAN
SELECT "title" FROM "movies" WHERE "id" IN (  SELECT "movie_id" FROM "stars" WHERE "person_id" = (    SELECT "id" FROM "people" WHERE "name" = 'Tom Hanks'  )  );
-- Compare runtime with indexes
.timer on
SELECT "title" FROM "movies" WHERE "id" IN (  
    SELECT "movie_id" FROM "stars" WHERE "person_id" IN (  
        SELECT "id" FROM "people" WHERE "name" = 'Tom Hanks'
    )  
)
.timer off
-- Demonstrates partial indexes
-- movies.db

-- Time searching for movies in 2023
.timer on
SELECT "title" FROM "movies" WHERE "year" = 2023;

-- Create a partial index to speed up searches involving the present year
CREATE INDEX "recents" ON "movies" ("title") WHERE "year" = 2023;

-- Rerun query
SELECT "title" FROM "movies" WHERE "year" = 2023;

-- Show query's usage of index
EXPLAIN QUERY PLAN
SELECT "title" FROM "movies" WHERE "year" = 2023;

-- Show not using an index after creating a partial index
EXPLAIN QUERY PLAN
SELECT "title" FROM "movies" WHERE "year" = 1998;
-- Demonstrates vacuum to reclaim unused space
-- movies.db

-- Drop existing indexes
DROP INDEX IF EXISTS "title_index";
DROP INDEX IF EXISTS "people_index";
DROP INDEX IF EXISTS "name_index";

-- Run vacuum to reclaim space
VACUUM;
-- Demonstrates single-column indexes
-- movies.db

-- Show schema of movies.db
.schema

-- Peek at movies table
SELECT * FROM "movies" LIMIT 5;

-- Search for a movie with a unique entry
SELECT * FROM "movies" WHERE "title" = 'Cars';

-- Search again, save terminal window
.timer on
SELECT * FROM "movies" WHERE "title" = 'Cars';
.timer off

-- Open new terminal window, create index on titles column
.timer on
CREATE INDEX "title_index" ON "movies" ("title");
.timer off

-- Show index as part of schema
.schema

-- Search again, via index
.timer on
SELECT * FROM "movies" WHERE "title" = 'Cars';
.timer off

-- Use EXPLAIN QUERY PLAN to show use of index
EXPLAIN QUERY PLAN
SELECT * FROM "movies" WHERE "title" = 'Cars';

-- DROP INDEX
DROP INDEX "title_index";

-- Show query plan without index
EXPLAIN QUERY PLAN
SELECT * FROM "movies" WHERE "title" = 'Cars';