```sql
-- Demonstrates race conditions
-- Uses bank.db

-- 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;
```

```sql
-- Demonstrates atomicity of transactions
-- Uses bank.db

-- Shows schema, higlight CHECK constraint
.schema

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

-- Updates 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

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

-- Creates a transaction which is successful
-- Views 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;

-- Completes 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

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

-- Creates 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;
```

```sql
-- Demonstrates foreign key indexes
-- Uses movies.db

-- Times 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

-- Identifies 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'
    )
);

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

-- Creates index to speed name lookups
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'
    )
);

-- Makes "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'
    )
);

-- Finds runtime with usage of 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
```

```sql
-- Demonstrates partial indexes
-- Uses movies.db

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

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

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

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

-- Shows not using an index after creating a partial index
EXPLAIN QUERY PLAN
SELECT "title" FROM "movies" WHERE "year" = 1998;
```

```sql
-- Demonstrates vacuum to reclaim unused space
-- Uses movies.db

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

-- Runs vacuum to reclaim space
VACUUM;
```

```sql
-- Demonstrates single-column indexes
-- Uses movies.db

-- Shows schema of movies.db
.schema

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

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

-- Searches again, with timer
.timer on
SELECT * FROM "movies" WHERE "title" = 'Cars';
.timer off

-- Creates index on titles column, with timer
.timer on
CREATE INDEX "title_index" ON "movies" ("title");
.timer off

-- Shows index as part of schema
.schema

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

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

-- Deletes "title_index" index
DROP INDEX "title_index";

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