

```
1  -- Demonstrates race conditions
2  -- Uses bank.db
3
4  -- Connection 1
5  SELECT "balance" FROM "accounts" WHERE "id" = 3;
6
7  -- Connection 2
8  SELECT "balance" FROM "accounts" WHERE "id" = 3;
9
10 -- Connection 1
11 UPDATE "accounts" SET "balance" = "balance" + 30 WHERE "id" = 1;
12
13 -- Connection 2
14 UPDATE "accounts" SET "balance" = "balance" + 30 WHERE "id" = 1;
15
16 -- Connection 3
17 SELECT "balance" FROM "accounts" WHERE "id" = 1;
18 UPDATE "accounts" SET "balance" = "balance" - 30 WHERE "id" = 1;
19
20 -- Connection 1
21 UPDATE "accounts" SET "balance" = "balance" - 30 WHERE "id" = 3;
22
23 -- Connection 2
24 UPDATE "accounts" SET "balance" = "balance" - 30 WHERE "id" = 3;
```

```
1  -- Demonstrates atomicity of transactions
2  -- Uses bank.db
3
4  -- Shows schema, highlight CHECK constraint
5  .schema
6
7  -- Views account balances
8  SELECT * FROM "accounts";
9
10 -- Updates balance without a transaction
11 UPDATE "accounts" SET "balance" = "balance" + 10 WHERE "id" = 2;
12 SELECT * FROM "accounts"; -- Viewing here provides an improper view of total balances
13 UPDATE "accounts" SET "balance" = "balance" - 10 WHERE "id" = 1;
14 SELECT * FROM "accounts"; -- Viewing here, after all updated, results in proper view
15
16 -- Resets table
17 UPDATE "accounts" SET "balance" = "balance" - 10 WHERE "id" = 2;
18 UPDATE "accounts" SET "balance" = "balance" + 10 WHERE "id" = 1;
19
20 -- Creates a transaction which is successful
21 -- Views state of database from other terminal mid-way through transaction
22 BEGIN TRANSACTION;
23 UPDATE "accounts" SET "balance" = "balance" + 10 WHERE "id" = 2;
24 UPDATE "accounts" SET "balance" = "balance" - 10 WHERE "id" = 1;
25 COMMIT;
26
27 -- Completes invalid update of balance without a transaction
28 UPDATE "accounts" SET "balance" = "balance" + 10 WHERE "id" = 2;
29 UPDATE "accounts" SET "balance" = "balance" - 10 WHERE "id" = 1; -- Invokes constraint error, which is ABORTed
30
31 -- Rolls back the balance
32 UPDATE "accounts" SET "balance" = "balance" - 10 WHERE "id" = 2;
33
34 -- Creates a transaction which should be rolled back
35 BEGIN TRANSACTION;
36 UPDATE "accounts" SET "balance" = "balance" + 10 WHERE "id" = 2;
37 UPDATE "accounts" SET "balance" = "balance" - 10 WHERE "id" = 1; -- Invokes constraint error, which is ABORTed
38 ROLLBACK;
```

```
1  -- Demonstrates foreign key indexes
2  -- Uses movies.db
3
4  -- Times searching for movies Tom Hanks has starred in
5  .timer on
6  SELECT "title" FROM "movies" WHERE "id" IN (
7      SELECT "movie_id" FROM "stars" WHERE "person_id" = (
8          SELECT "id" FROM "people" WHERE "name" = 'Tom Hanks'
9      )
10 );
11 .timer off
12
13 -- Identifies which columns we should create indexes on
14 EXPLAIN QUERY PLAN
15 SELECT "title" FROM "movies" WHERE "id" IN (
16     SELECT "movie_id" FROM "stars" WHERE "person_id" = (
17         SELECT "id" FROM "people" WHERE "name" = 'Tom Hanks'
18     )
19 );
20
21 -- Creates index on foreign key
22 .timer on
23 CREATE INDEX "person_index" ON "stars" ("person_id");
24
25 -- Creates index to speed name lookups
26 CREATE INDEX "name_index" ON "people" ("name");
27 .timer off
28
29 EXPLAIN QUERY PLAN
30 SELECT "title" FROM "movies" WHERE "id" IN (
31     SELECT "movie_id" FROM "stars" WHERE "person_id" = (
32         SELECT "id" FROM "people" WHERE "name" = 'Tom Hanks'
33     )
34 );
35
36 -- Makes "person_index" a covering index for the above query
37 CREATE INDEX "person_index" ON "stars" ("person_id", "movie_id");
38
39 EXPLAIN QUERY PLAN
40 SELECT "title" FROM "movies" WHERE "id" IN (
41     SELECT "movie_id" FROM "stars" WHERE "person_id" = (
42         SELECT "id" FROM "people" WHERE "name" = 'Tom Hanks'
```

```
43     )
44 );
45
46 -- Finds runtime with usage of indexes
47 .timer on
48 SELECT "title" FROM "movies" WHERE "id" IN (
49     SELECT "movie_id" FROM "stars" WHERE "person_id" IN (
50         SELECT "id" FROM "people" WHERE "name" = 'Tom Hanks'
51     )
52 );
53 .timer off
```

```
1  -- Demonstrates partial indexes
2  -- Uses movies.db
3
4  -- Times searching for movies in 2023
5  .timer on
6  SELECT "title" FROM "movies" WHERE "year" = 2023;
7
8  -- Creates a partial index to speed up searches involving the present year
9  CREATE INDEX "recents" ON "movies" ("title")
10 WHERE "year" = 2023;
11
12 -- Reruns query
13 SELECT "title" FROM "movies" WHERE "year" = 2023;
14
15 -- Shows query's usage of index
16 EXPLAIN QUERY PLAN
17 SELECT "title" FROM "movies" WHERE "year" = 2023;
18
19 -- Shows not using an index after creating a partial index
20 EXPLAIN QUERY PLAN
21 SELECT "title" FROM "movies" WHERE "year" = 1998;
```

```
1  -- Demonstrates vacuum to reclaim unused space
2  -- Uses movies.db
3
4  -- Drops existing indexes
5  DROP INDEX IF EXISTS "title_index";
6  DROP INDEX IF EXISTS "people_index";
7  DROP INDEX IF EXISTS "name_index";
8
9  -- Runs vacuum to reclaim space
10 VACUUM;
```

```
1  -- Demonstrates single-column indexes
2  -- Uses movies.db
3
4  -- Shows schema of movies.db
5  .schema
6
7  -- Peeks at movies table
8  SELECT * FROM "movies" LIMIT 5;
9
10 -- Searches for a movie with a unique entry
11 SELECT * FROM "movies" WHERE "title" = 'Cars';
12
13 -- Searches again, with timer
14 .timer on
15 SELECT * FROM "movies" WHERE "title" = 'Cars';
16 .timer off
17
18 -- Creates index on titles column, with timer
19 .timer on
20 CREATE INDEX "title_index" ON "movies" ("title");
21 .timer off
22
23 -- Shows index as part of schema
24 .schema
25
26 -- Searches again, via index, with timer
27 .timer on
28 SELECT * FROM "movies" WHERE "title" = 'Cars';
29 .timer off
30
31 -- Uses EXPLAIN QUERY PLAN to show use of index
32 EXPLAIN QUERY PLAN
33 SELECT * FROM "movies" WHERE "title" = 'Cars';
34
35 -- Deletes "title_index" index
36 DROP INDEX "title_index";
37
38 -- Shows query plan without index
39 EXPLAIN QUERY PLAN
40 SELECT * FROM "movies" WHERE "title" = 'Cars';
```