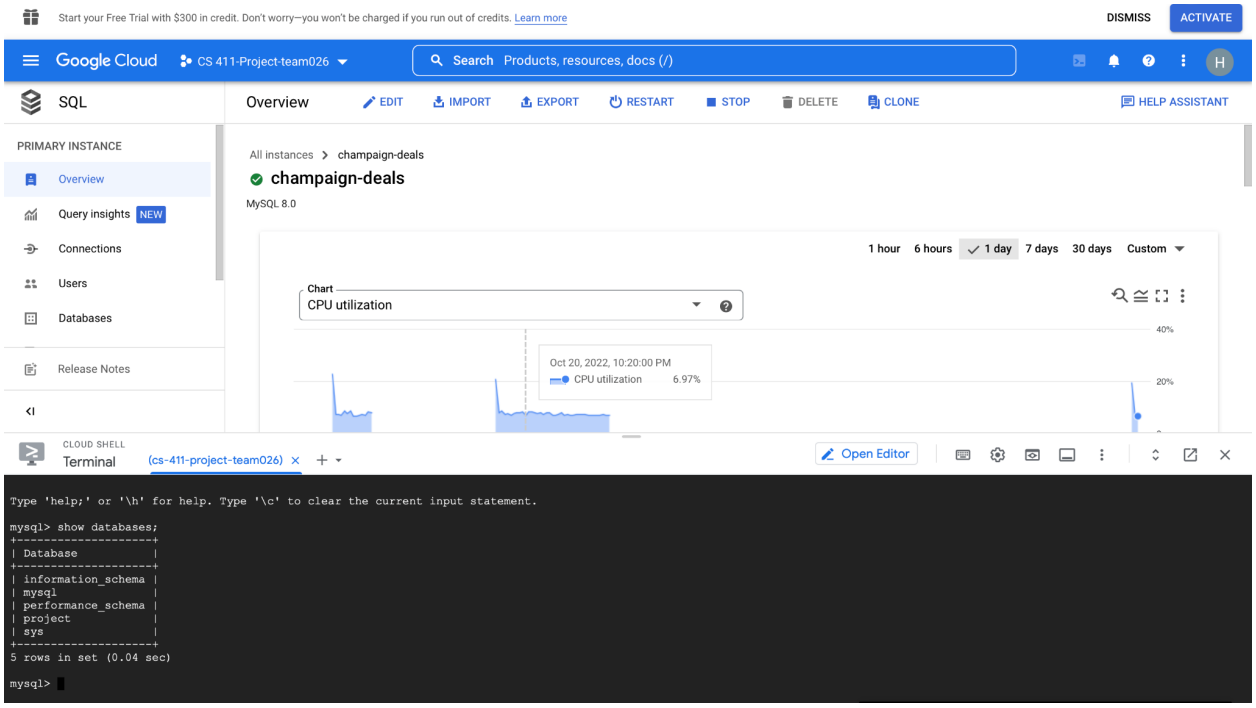
Database Design

Implementing the Database Table on GCP



Data Definition Language (DDL)

User

```
CREATE TABLE User (
  id INTEGER NOT NULL PRIMARY KEY,
  name TEXT,
  description TEXT,
  email TEXT
);
```

Category

```
CREATE TABLE Category (
  id INTEGER NOT NULL PRIMARY KEY,
  name TEXT
);
```

```
CREATE TABLE Location (
  id INTEGER NOT NULL PRIMARY KEY,
  name TEXT
);
```

Item

```
CREATE TABLE Item (
  id INTEGER NOT NULL PRIMARY KEY,
  name TEXT,
  description TEXT,
  price REAL,
  publishDate DATE,
  status VARCHAR(20),
  locationId INTEGER NOT NULL,
  categoryId INTEGER NOT NULL,
  FOREIGN KEY (locationId) REFERENCES Location(id),
  FOREIGN KEY (categoryId) REFERENCES Category(id)
);
```

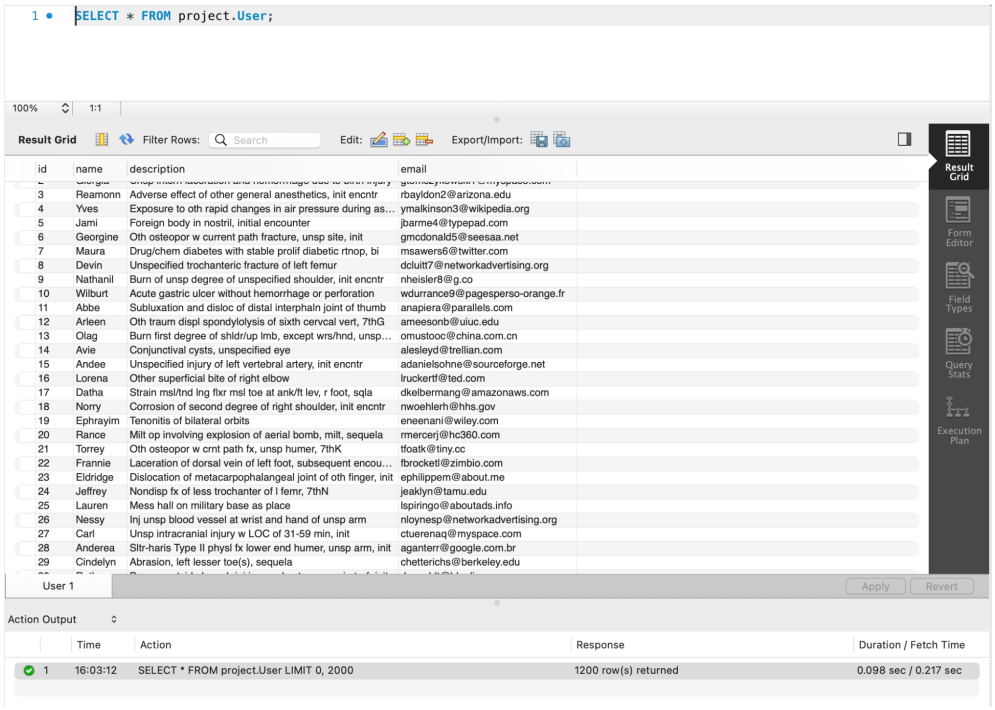
Comment

```
CREATE TABLE Comment (
  id INTEGER NOT NULL PRIMARY KEY,
  content TEXT,
  date DATE,
  itemId INTEGER NOT NULL,
  FOREIGN KEY (itemId) REFERENCES Item(id)
);
```

Transaction

```
CREATE TABLE Transaction (
  sellerId INTEGER NOT NULL,
  itemId INTEGER NOT NULL,
  buyerId INTEGER NOT NULL,
  status VARCHAR(20),
  note TEXT,
  PRIMARY KEY (sellerId, itemId, buyerId),
  FOREIGN KEY (sellerId) REFERENCES User(id),
  FOREIGN KEY (buyerId) REFERENCES User(id),
  FOREIGN KEY (itemId) REFERENCES Item(id)
);
```

Inserting at least 1000 Rows



```
1 • SELECT * FROM project.User;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import:

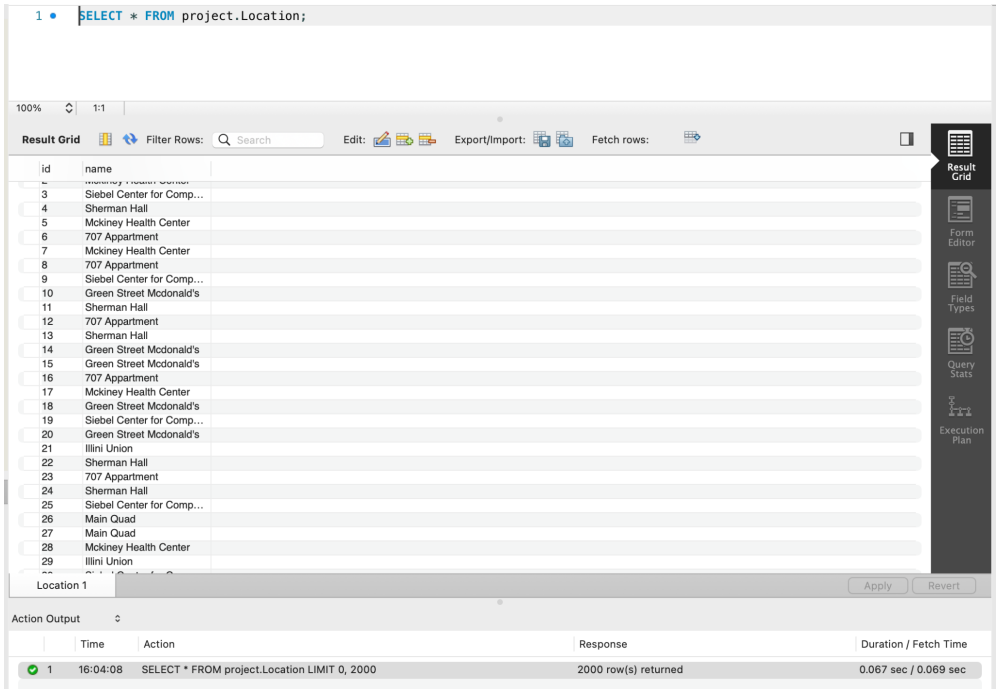
id	name	description	email
3	Reamonn	Adverse effect of other general anesthetics, init encntr	rbayldon2@arizona.edu
4	Yves	Exposure to oth rapid changes in air pressure during as…	ymalkinson3@wikipedia.org
5	Jami	Foreign body in nostril, initial encounter	jbarme4@typepad.com
6	Georgine	Oth osteopor w current path fracture, unsp site, init	gmcdonald5@seesaa.net
7	Maura	Drug/chem diabetes with stable prolif diabetic rtnop, bi	msawers6@twitter.com
8	Devin	Unspecified trochanteric fracture of left femur	dcluitt7@networkadvertising.org
9	Nathanil	Burn of unsp degree of unspecified shoulder, init encntr	nheisler8@g.co
10	Wilburt	Acute gastric ulcer without hemorrhage or perforation	wdurrance9@pagesperso-orange.fr
11	Abbe	Subluxation and disloc of distal interphaln joint of thumb	anapiera@parallels.com
12	Arleen	Oth traum displ spondylolysis of sixth cervcal vert, 7thG	ameesonb@uiuc.edu
13	Olag	Burn first degree of shldr/up lmb, except wrs/hnd, unsp…	omustooc@china.com.cn
14	Avie	Conjunctival cysts, unspecified eye	alesleyd@trellian.com
15	Andee	Unspecified injury of left vertebral artery, init encntr	adanielsohne@sourceforge.net
16	Lorena	Other superficial bite of right elbow	lruckertf@ted.com
17	Datha	Strain msl/tnd lng flxr msl toe at ank/ft lev, r foot, sqla	dkelbermang@amazonaws.com
18	Norry	Corrosion of second degree of right shoulder, init encntr	nwoehlerh@hhs.gov
19	Ephrayim	Tenonitis of bilateral orbits	eneenani@wiley.com
20	Rance	Milt op involving explosion of aerial bomb, milt, sequela	rmercerj@hc360.com
21	Torrey	Oth osteopor w crnt path fx, unsp humer, 7thK	tfoatk@tiny.cc
22	Frannie	Laceration of dorsal vein of left foot, subsequent encou…	fbrocketl@zimbio.com
23	Eldridge	Dislocation of metacarpophalangeal joint of oth finger, init	ephilippem@about.me
24	Jeffrey	Nondisp fx of less trochanter of l femr, 7thN	jeaklyn@tamu.edu
25	Lauren	Mess hall on military base as place	lspiringo@aboutads.info
26	Nessy	Inj unsp blood vessel at wrist and hand of unsp arm	nloynesp@networkadvertising.org
27	Carl	Unsp intracranial injury w LOC of 31-59 min, init	ctuerenaq@myspace.com
28	Anderea	Sltr-haris Type II physl fx lower end humer, unsp arm, init	aganterr@google.com.br
29	Cindelyn	Abrasion, left lesser toe(s), sequela	chetterichs@berkeley.edu

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

User 1 Apply Revert

Action Output

	Time	Action	Response	Duration / Fetch Time
1	16:03:12	SELECT * FROM project.User LIMIT 0, 2000	1200 row(s) returned	0.098 sec / 0.217 sec



```
1 • SELECT * FROM project.Location;
```

100% 1:1

Result Grid Filter Rows: Search Edit: Export/Import: Fetch rows:

id	name
3	Siebel Center for Comp…
4	Sherman Hall
5	Mckiney Health Center
6	707 Appartment
7	Mckiney Health Center
8	707 Appartment
9	Siebel Center for Comp…
10	Green Street Mcdonald's
11	Sherman Hall
12	707 Appartment
13	Sherman Hall
14	Green Street Mcdonald's
15	Green Street Mcdonald's
16	707 Appartment
17	Mckiney Health Center
18	Green Street Mcdonald's
19	Siebel Center for Comp…
20	Green Street Mcdonald's
21	Illini Union
22	Sherman Hall
23	707 Appartment
24	Sherman Hall
25	Siebel Center for Comp…
26	Main Quad
27	Main Quad
28	Mckiney Health Center
29	Illini Union

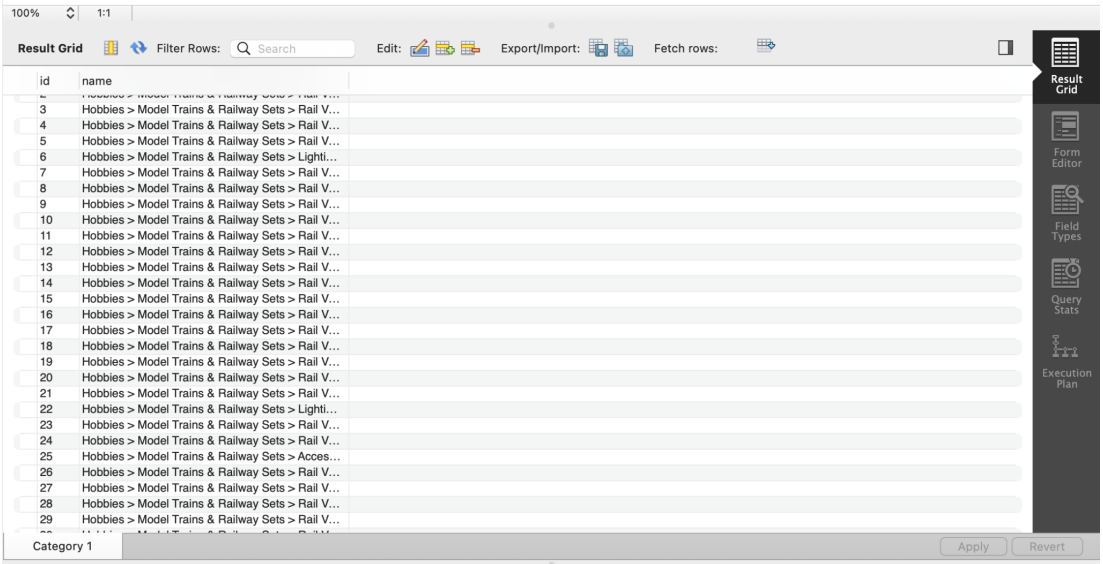
Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Location 1 Apply Revert

Action Output

	Time	Action	Response	Duration / Fetch Time
1	16:04:08	SELECT * FROM project.Location LIMIT 0, 2000	2000 row(s) returned	0.067 sec / 0.069 sec

```
1 • SELECT * FROM project.Category;
```



Result Grid | Filter Rows: Search | Edit: | Export/Import: | Fetch rows:

id	name
3	Hobbies > Model Trains & Railway Sets > Rail V…
4	Hobbies > Model Trains & Railway Sets > Rail V…
5	Hobbies > Model Trains & Railway Sets > Rail V…
6	Hobbies > Model Trains & Railway Sets > Lighti…
7	Hobbies > Model Trains & Railway Sets > Rail V…
8	Hobbies > Model Trains & Railway Sets > Rail V…
9	Hobbies > Model Trains & Railway Sets > Rail V…
10	Hobbies > Model Trains & Railway Sets > Rail V…
11	Hobbies > Model Trains & Railway Sets > Rail V…
12	Hobbies > Model Trains & Railway Sets > Rail V…
13	Hobbies > Model Trains & Railway Sets > Rail V…
14	Hobbies > Model Trains & Railway Sets > Rail V…
15	Hobbies > Model Trains & Railway Sets > Rail V…
16	Hobbies > Model Trains & Railway Sets > Rail V…
17	Hobbies > Model Trains & Railway Sets > Rail V…
18	Hobbies > Model Trains & Railway Sets > Rail V…
19	Hobbies > Model Trains & Railway Sets > Rail V…
20	Hobbies > Model Trains & Railway Sets > Rail V…
21	Hobbies > Model Trains & Railway Sets > Rail V…
22	Hobbies > Model Trains & Railway Sets > Lighti…
23	Hobbies > Model Trains & Railway Sets > Rail V…
24	Hobbies > Model Trains & Railway Sets > Rail V…
25	Hobbies > Model Trains & Railway Sets > Acces…
26	Hobbies > Model Trains & Railway Sets > Rail V…
27	Hobbies > Model Trains & Railway Sets > Rail V…
28	Hobbies > Model Trains & Railway Sets > Rail V…
29	Hobbies > Model Trains & Railway Sets > Rail V…

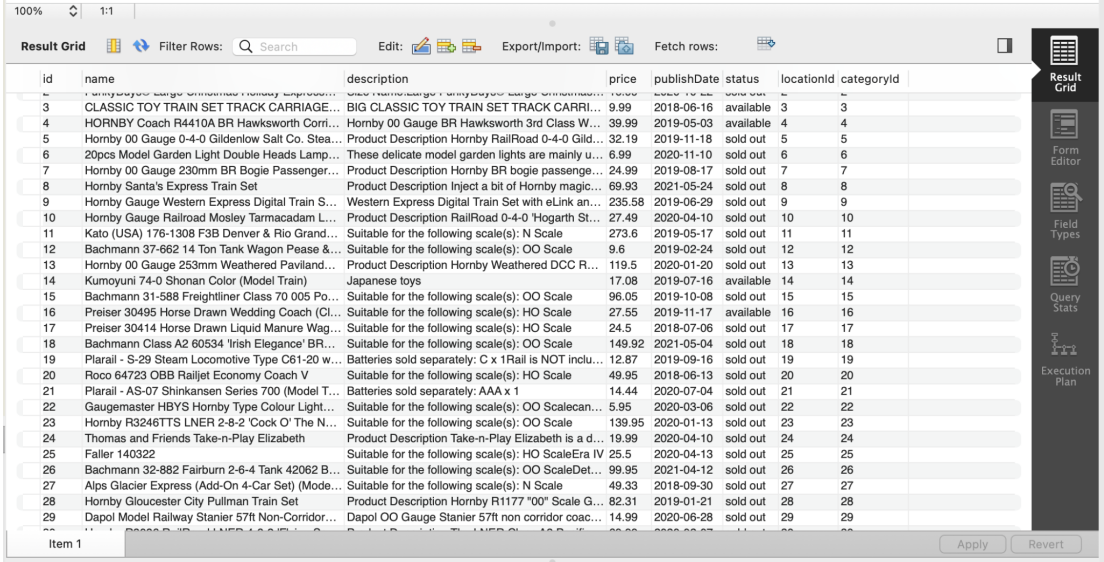
Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Category 1 | Apply | Revert

Action Output

	Time	Action	Response	Duration / Fetch Time
1	16:04:50	SELECT * FROM project.Category LIMIT 0, 2000	2000 row(s) returned	0.076 sec / 0.120 sec

```
1 • SELECT * FROM project.Item;
```



Result Grid | Filter Rows: Search | Edit: | Export/Import: | Fetch rows:

id	name	description	price	publishDate	status	locationId	categoryId
3	CLASSIC TOY TRAIN SET TRACK CARRIAGE…	BIG CLASSIC TOY TRAIN SET TRACK CARRI…	9.99	2018-06-16	available	3	3
4	HORNBY Coach R4410A BR Hawksworth Corri…	Hornby 00 Gauge BR Hawksworth 3rd Class W…	39.99	2019-05-03	available	4	4
5	Hornby 00 Gauge 0-4-0 Gildenlow Salt Co. Stea…	Product Description Hornby RailRoad 0-4-0 Gild…	32.19	2019-11-18	sold out	5	5
6	20pcs Model Garden Light Double Heads Lamp…	These delicate model garden lights are mainly u…	6.99	2020-11-10	sold out	6	6
7	Hornby 00 Gauge 230mm BR Bogie Passenger…	Product Description Hornby BR bogie passenge…	24.99	2019-08-17	sold out	7	7
8	Hornby Santa's Express Train Set	Product Description Inject a bit of Hornby magic…	69.93	2021-05-24	sold out	8	8
9	Hornby Gauge Western Express Digital Train S…	Western Express Digital Train Set with eLink an…	235.58	2019-06-29	sold out	9	9
10	Hornby Gauge Railroad Mosley Tarmacadam L…	Product Description RailRoad 0-4-0 'Hogarth St…	27.49	2020-04-10	sold out	10	10
11	Kato (USA) 176-1308 F3B Denver & Rio Grand…	Suitable for the following scale(s): N Scale	273.6	2019-05-17	sold out	11	11
12	Bachmann 37-662 14 Ton Tank Wagon Pease &…	Suitable for the following scale(s): OO Scale	9.6	2019-02-24	sold out	12	12
13	Hornby 00 Gauge 253mm Weathered Paviland…	Product Description Hornby Weathered DCC R…	119.5	2020-01-20	sold out	13	13
14	Kumoyuni 74-0 Shonan Color (Model Train)	Japanese toys	17.08	2019-07-16	available	14	14
15	Bachmann 31-588 Freightliner Class 70 005 Po…	Suitable for the following scale(s): OO Scale	96.05	2019-10-08	sold out	15	15
16	Preiser 30495 Horse Drawn Wedding Coach (Cl…	Suitable for the following scale(s): HO Scale	27.55	2019-11-17	available	16	16
17	Preiser 30414 Horse Drawn Liquid Manure Wag…	Suitable for the following scale(s): HO Scale	24.5	2018-07-06	sold out	17	17
18	Bachmann Class A2 60534 'Irish Elegance' BR…	Suitable for the following scale(s): OO Scale	149.92	2021-05-04	sold out	18	18
19	Plarail - S-29 Steam Locomotive Type C61-20 w…	Batteries sold separately: C x 1Rail is NOT inclu…	12.87	2019-09-16	sold out	19	19
20	Roco 64723 OBB Railjet Economy Coach V	Suitable for the following scale(s): HO Scale	49.95	2018-06-13	sold out	20	20
21	Plarail - AS-07 Shinkansen Series 700 (Model T…	Batteries sold separately: AAA x 1	14.44	2020-07-04	sold out	21	21
22	Gaugemaster HBYS Hornby Type Colour Light…	Suitable for the following scale(s): OO Scalecan…	5.95	2020-03-06	sold out	22	22
23	Hornby R3246TTS LNER 2-8-2 'Cock O' The N…	Suitable for the following scale(s): OO Scale	139.95	2020-01-13	sold out	23	23
24	Thomas and Friends Take-n-Play Elizabeth	Product Description Take-n-Play Elizabeth is a d…	19.99	2020-04-10	sold out	24	24
25	Faller 140322	Suitable for the following scale(s): HO ScaleEra IV	25.5	2020-04-13	sold out	25	25
26	Bachmann 32-882 Fairburn 2-6-4 Tank 42062 B…	Suitable for the following scale(s): OO ScaleDet…	99.95	2021-04-12	sold out	26	26
27	Alps Glacier Express (Add-On 4-Car Set) (Mode…	Suitable for the following scale(s): N Scale	49.33	2018-09-30	sold out	27	27
28	Hornby Gloucester City Pullman Train Set	Product Description Hornby R1177 "00" Scale G…	82.31	2019-01-21	sold out	28	28
29	Dapol Model Railway Stanier 57ft Non-Corridor…	Dapol OO Gauge Stanier 57ft non corridor coac…	14.99	2020-06-28	sold out	29	29

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Item 1 | Apply | Revert

Action Output

	Time	Action	Response	Duration / Fetch Time
1	16:06:01	SELECT * FROM project.Item LIMIT 0, 2000	2000 row(s) returned	0.125 sec / 0.621 sec

```
1 • SELECT * FROM project.Comment;
```

100% 1:1

Result Grid | Filter Rows: Search | Edit: | Export/Import: | Fetch rows:

id	content	date	itemId
3	Lacerat extensor musc/fasc/tend l idx fngr at wr…	2018-10-19	3
4	Unsp injury of unsp msl/tnd at ank/ft level, unsp…	2020-08-17	4
5	Fx unsp part of scapula, unsp shldr, subs for fx…	2019-04-10	5
6	Preterm labor w preterm delivery, unsp trimester…	2020-10-03	6
7	Abnormal serum enzyme level, unspecified	2021-07-03	7
8	Corrosion of second degree of right hand, unsp…	2019-07-14	8
9	Cont preg aft elctv fetal rdct of one fetus or mor…	2018-05-14	9
10	Toxic effect of carbon monoxide from unsp sour…	2020-06-05	10
11	Epiphora	2019-10-24	11
12	Lacerat musc/tend ant grp at low leg level, right…	2018-08-31	12
13	Other injury of ovary, bilateral, initial encounter	2019-01-07	13
14	Other fracture of fifth metacarpal bone, right hand	2020-02-05	14
15	Carcinoid syndrome	2021-12-21	15
16	Nondisp artic fx head of unsp femr, 7thR	2019-10-21	16
17	Nondisp transverse fx r patella, 7thR	2021-07-07	17
18	Toxic effect of harmful algae and algae toxins, u…	2018-01-24	18
19	Cannabis use, unspecified with intoxication	2020-07-13	19
20	Unsp injury of musc/fasc/tend long hd bicep, un…	2020-09-10	20
21	Other congenital malformations of urinary system	2020-12-28	21
22	Nondisp fx of glenoid cav of scapula, unsp shldr…	2019-05-13	22
23	Epilepsy, unsp, not intractable, with status epile…	2019-03-30	23
24	Glaucoma secondary to eye trauma, bilateral, m…	2020-03-23	24
25	Hypertrophy of tonsils with hypertrophy of aden…	2019-07-30	25
26	Nondisp fx of unsp ulna styloid pro, 7thM	2021-07-15	26
27	Connective tissue stenosis of neural canal of sa…	2020-07-13	27
28	Oth injury of deep palmar arch of left hand, init e…	2021-05-25	28
29	Other kyphosis	2018-10-01	29

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Comment 1 | Apply | Revert

Action Output

	Time	Action	Response	Duration / Fetch Time
1	16:07:20	SELECT * FROM project.Comment LIMIT 0, 2000	2000 row(s) returned	0.086 sec / 0.056 sec

```
1 • SELECT * FROM project.Transaction;
```

100% 1:1

Result Grid | Filter Rows: Search | Edit: | Export/Import: | Fetch rows:

sellerId	itemId	buyerId	status	note
1	350	1133	completed	n/a
1	1260	937	completed	n/a
2	984	1038	completed	n/a
2	1527	189	completed	n/a
3	1447	1148	completed	n/a
4	453	1170	completed	n/a
4	887	862	completed	n/a
5	835	735	completed	n/a
6	626	704	completed	n/a
6	1771	252	completed	n/a
7	65	240	completed	n/a
7	1981	290	completed	n/a
8	974	366	completed	n/a
9	1076	224	completed	n/a
9	1633	753	completed	n/a
10	229	1075	completed	n/a
10	471	975	completed	n/a
11	389	147	completed	n/a
11	1338	128	completed	n/a
12	1969	528	completed	n/a
13	878	887	completed	n/a
13	1194	796	completed	n/a
14	153	576	completed	n/a
14	1409	1025	completed	n/a
15	1861	762	completed	n/a
15	1932	699	completed	n/a
16	1468	720	completed	n/a

Result Grid | Form Editor | Field Types | Query Stats | Execution Plan

Transaction 1 | Apply | Revert

Action Output

	Time	Action	Response	Duration / Fetch Time
1	16:07:39	SELECT * FROM project.Transaction LIMIT 0, 2000	2000 row(s) returned	0.099 sec / 0.070 sec

Advance Query

1. FIND GOOD SELLER

Filter out sellers who have published items since 2020 or have more than 3 comments in at least one item. Sort based on the sum price of all items each seller has sold and get the top ones in the list as a good seller.

```
(SELECT u.id , SUM(Price) as turnover
FROM User u JOIN Transaction T ON (u.id = T.sellerId) JOIN Item I
USING (id)
WHERE EXISTS(
    SELECT *
    FROM User u2 JOIN Item I2  USING (id)
    WHERE publishDate >= 2020 AND u2.id = u.id AND T.status =
"completed"
)
GROUP BY u.id
)

UNION

(SELECT u.id , SUM(Price) as turnover
FROM User u JOIN Item I  USING (id) JOIN Transaction T ON (I.id =
T.sellerId)
WHERE EXISTS(
     SELECT T.sellerId
     FROM Item I2 JOIN Comment c ON (I2.id = c.id) JOIN Transaction
T ON (I2.id = T.sellerId)
    WHERE T.status = "completed"
     GROUP BY T.sellerId
     HAVING COUNT(*) > 3
)
GROUP BY u.id)
ORDER BY turnover DESC
LIMIT 15;
```

	id	turnover
▶	159	1077.46
	533	706.88
	727	558
	11	547.2
	9	471.16
	692	460.22
	575	457.9
	426	457.9
	614	455.36
	434	453.14
	643	394.84
	1122	379.89
	114	339.98
	1145	337
	47	334.2

2. FIND CHEAP Category

Filter based on each category and find the number of available items in each category that has prices less than 5 Euro. Sort out by descending order of the number of items each category satisfied that requirement.

```
SELECT c.name, COUNT(*) AS Num_Item
FROM Category c JOIN (
      SELECT i.id, i.categoryId FROM Item i
     WHERE i.price < 5 AND i.status = 'available'
) as subtable ON c.id = subtable.categoryId
GROUP BY c.name
ORDER BY Num_Item DESC
LIMIT 15;
```

name	Num_Item
Arts & Crafts > Paper & Stickers	27
Arts & Crafts > Children's Craft Kits > Bead Art…	18
Party Supplies > Decorations > Balloons	17
Games > Dice & Dice Games	15
Fancy Dress > Accessories > Temporary Tattoos	14
Hobbies > Trading Cards & Accessories > Pack…	8
Hobbies > Remote Controlled Devices > Parts…	5
Characters & Brands > Disney > Toys	4
Arts & Crafts > Art Sand	3
Sports Toys & Outdoor > Beach Toys > Flotation…	3
Die-Cast & Toy Vehicles > Toy Vehicles & Acces…	2
Characters & Brands > Toy Story > Toys	2
Figures & Playsets > Science Fiction & Fantasy	2
Party Supplies > Banners Stickers & Confetti >…	2
Hobbies > Model Trains & Railway Sets > Lighti…	1

Indexing

1. FIND GOOD SELLER

Original

```
-> Sort: turnover DESC  (cost=2.50 rows=0) (actual time=0.505..0.600 rows=1200 loops=1)
    -> Table scan on <union temporary>  (cost=2.50 rows=0) (actual time=0.001..0.079 rows=1200 loops=1)
        -> Union materialize with deduplication  (cost=2.50..2.50 rows=0) (actual time=18.936..19.130 rows=1200 loops=1)
            -> Table scan on <temporary>  (actual time=0.001..0.079 rows=1200 loops=1)
                -> Aggregate using temporary table  (actual time=17.849..18.019 rows=1200 loops=1)
                    -> Nested loop inner join  (cost=483.25 rows=67) (actual time=0.099..16.488 rows=2000 loops=1)
                        -> Nested loop inner join  (cost=413.25 rows=200) (actual time=0.088..13.706 rows=2000 loops=1)
                            -> Nested loop inner join  (cost=343.25 rows=200) (actual time=0.080..11.222 rows=2000 loops=1)
                                -> Nested loop inner join  (cost=273.25 rows=200) (actual time=0.075..4.383 rows=2000 loops=1)
                                    -> Filter: (T.`status` = 'completed')  (cost=203.25 rows=200) (actual time=0.064..2.087 rows=2000 loops=1)
                                        -> Table scan on T  (cost=203.25 rows=2000) (actual time=0.061..1.202 rows=2000 loops=1)
                                    -> Single-row index lookup on u using PRIMARY (id=T.sellerId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=2000)
                                -> Single-row index lookup on u2 using PRIMARY (id=T.sellerId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=2000)
                            -> Single-row index lookup on I using PRIMARY (id=T.sellerId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=2000)
                        -> Filter: (I2.publishDate >= 2020)  (cost=0.25 rows=0) (actual time=0.001..0.001 rows=1 loops=2000)
                            -> Single-row index lookup on I2 using PRIMARY (id=T.sellerId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=2000)
            -> Zero rows (Impossible WHERE)  (cost=0.00..0.00 rows=0) (actual time=0.000..0.000 rows=0 loops=1)

```

CREATE INDEX STATUS
ON Transaction(status)

(Revert: ALTER TABLE Transaction
DROP INDEX STATUS)

```
-> Sort: turnover DESC  (cost=2.50 rows=0) (actual time=0.337..0.401 rows=1200 loops=1)
    -> Table scan on <union temporary>  (cost=2.50 rows=0) (actual time=0.002..0.058 rows=1200 loops=1)
        -> Union materialize with deduplication  (cost=1682.47..1684.97 rows=667) (actual time=9.225..9.362 rows=1200 loops=1)
            -> Group aggregate: sum(I.price)  (cost=1615.81 rows=667) (actual time=0.066..8.485 rows=1200 loops=1)
                -> Nested loop inner join  (cost=1549.15 rows=667) (actual time=0.053..7.819 rows=2000 loops=1)
                    -> Nested loop inner join  (cost=1382.50 rows=400) (actual time=0.040..4.490 rows=1200 loops=1)
                        -> Nested loop inner join  (cost=962.50 rows=1200) (actual time=0.032..3.007 rows=1200 loops=1)
                            -> Nested loop inner join  (cost=542.50 rows=1200) (actual time=0.028..1.620 rows=1200 loops=1)
                                -> Index scan on u using PRIMARY  (cost=122.50 rows=1200) (actual time=0.023..0.307 rows=1200 loops=1)
                                -> Single-row index lookup on u2 using PRIMARY (id=u.id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1200)
                            -> Single-row index lookup on I using PRIMARY (id=u.id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1200)
                        -> Filter: (I2.publishDate >= 2020)  (cost=0.25 rows=0) (actual time=0.001..0.001 rows=1 loops=1200)
                            -> Single-row index lookup on I2 using PRIMARY (id=u.id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1200)
                    -> Filter: (T.`status` = 'completed')  (cost=0.25 rows=2) (actual time=0.002..0.003 rows=2 loops=1200)
                        -> Index lookup on T using PRIMARY (sellerId=u.id)  (cost=0.25 rows=2) (actual time=0.001..0.002 rows=2 loops=1200)
            -> Zero rows (Impossible WHERE)  (cost=0.00..0.00 rows=0) (actual time=0.000..0.000 rows=0 loops=1)

```

Explanation: We find that filtering status of transactions takes 0.064...2.087, which is a lot of time, so we decide to create an index on T.status. As a result, query time has been reduced.

CREATE INDEX PUBLISH_DATE
ON Item(publishDate)

(Revert:
ALTER TABLE Item
DROP INDEX PUBLISH_DATE)

```
-> Sort: turnover DESC  (cost=2.50 rows=0) (actual time=0.365..0.429 rows=1200 loops=1)
    -> Table scan on <union temporary>  (cost=2.50 rows=0) (actual time=0.002..0.055 rows=1200 loops=1)
        -> Union materialize with deduplication  (cost=1682.47..1684.97 rows=667) (actual time=9.854..9.986 rows=1200 loops=1)
            -> Group aggregate: sum(I.price)  (cost=1615.81 rows=667) (actual time=0.068..9.030 rows=1200 loops=1)
                -> Nested loop inner join  (cost=1549.15 rows=667) (actual time=0.054..8.296 rows=2000 loops=1)
                    -> Nested loop inner join  (cost=1382.50 rows=400) (actual time=0.041..4.665 rows=1200 loops=1)
                        -> Nested loop inner join  (cost=962.50 rows=1200) (actual time=0.033..3.290 rows=1200 loops=1)
                            -> Nested loop inner join  (cost=542.50 rows=1200) (actual time=0.028..1.824 rows=1200 loops=1)
                                -> Index scan on u using PRIMARY  (cost=122.50 rows=1200) (actual time=0.024..0.319 rows=1200 loops=1)
                                -> Single-row index lookup on u2 using PRIMARY (id=u.id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1200)
                            -> Single-row index lookup on I using PRIMARY (id=u.id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1200)
                        -> Filter: (I2.publishDate >= 2020)  (cost=0.25 rows=0) (actual time=0.001..0.001 rows=1 loops=1200)
                            -> Single-row index lookup on I2 using PRIMARY (id=u.id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1200)
                    -> Filter: (T.`status` = 'completed')  (cost=0.25 rows=2) (actual time=0.002..0.003 rows=2 loops=1200)
                        -> Index lookup on T using PRIMARY (sellerId=u.id)  (cost=0.25 rows=2) (actual time=0.002..0.002 rows=2 loops=1200)
            -> Zero rows (Impossible WHERE)  (cost=0.00..0.00 rows=0) (actual time=0.000..0.000 rows=0 loops=1)

```

Explanation: Since publish date is used in the query, we tried to add an index to it. However, the query time has not been reduced. We presume that this is because the database compares publish dates of entries one by one in actual execution as shown in the analysis, so the index is not referenced at all.

CREATE INDEX SELLER_ID
ON Transaction(sellerId)

(Revert:
ALTER TABLE Transaction
DROP INDEX SELLER_ID)

```
-> Sort: turnover DESC  (cost=2.50 rows=0) (actual time=0.336..0.399 rows=1200 loops=1)
    -> Table scan on <union temporary>  (cost=2.50 rows=0) (actual time=0.002..0.055 rows=1200 loops=1)
        -> Union materialize with deduplication  (cost=1682.47..1684.97 rows=667) (actual time=9.195..9.329 rows=1200 loops=1)
            -> Group aggregate: sum(I.price)  (cost=1615.81 rows=667) (actual time=0.071..8.459 rows=1200 loops=1)
                -> Nested loop inner join  (cost=1549.15 rows=667) (actual time=0.057..7.826 rows=2000 loops=1)
                    -> Nested loop inner join  (cost=1382.50 rows=400) (actual time=0.044..4.422 rows=1200 loops=1)
                        -> Nested loop inner join  (cost=962.50 rows=1200) (actual time=0.036..3.065 rows=1200 loops=1)
                            -> Nested loop inner join  (cost=542.50 rows=1200) (actual time=0.032..1.713 rows=1200 loops=1)
                                -> Index scan on u using PRIMARY  (cost=122.50 rows=1200) (actual time=0.026..0.317 rows=1200 loops=1)
                                -> Single-row index lookup on u2 using PRIMARY (id=u.id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1200)
                            -> Single-row index lookup on I using PRIMARY (id=u.id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1200)
                        -> Filter: (I2.publishDate >= 2020)  (cost=0.25 rows=0) (actual time=0.001..0.001 rows=1 loops=1200)
                            -> Single-row index lookup on I2 using PRIMARY (id=u.id)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=1200)
                    -> Filter: (T.`status` = 'completed')  (cost=0.25 rows=2) (actual time=0.002..0.003 rows=2 loops=1200)
                        -> Index lookup on T using PRIMARY (sellerId=u.id)  (cost=0.25 rows=2) (actual time=0.001..0.002 rows=2 loops=1200)
            -> Zero rows (Impossible WHERE)  (cost=0.00..0.00 rows=0) (actual time=0.000..0.000 rows=0 loops=1)

```

Explanation: Since seller ID is used in the query, we tried to add an index to it. However, the query time has not been reduced. We presume that this is because the database compares a constant number of entries of seller ID with user IDs in actual execution as shown in the analysis, so the index is not referenced at all.

2. FIND CHEAP Category

Original

```
-> Sort: Num_Item DESC  (actual time=1.622..1.629 rows=27 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.004 rows=27 loops=1)
        -> Aggregate using temporary table  (actual time=1.597..1.601 rows=27 loops=1)
            -> Nested loop inner join  (cost=254.06 rows=69) (actual time=0.088..1.385 rows=135 loops=1)
                -> Filter: ((i.`status` = 'available') and (i.price < 5))  (cost=230.05 rows=69) (actual time=0.074..1.161 rows=135 loops=1)
                    -> Table scan on i  (cost=230.05 rows=2058) (actual time=0.061..0.862 rows=2000 loops=1)
                -> Single-row index lookup on c using PRIMARY (id=i.categoryId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=135)

```

CREATE INDEX PRICE
ON Item(Price)

(Revert:
ALTER TABLE Item
DROP INDEX PRICE)

```
-> Sort: Num_Item DESC  (actual time=1.516..1.522 rows=27 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.003 rows=27 loops=1)
        -> Aggregate using temporary table  (actual time=1.495..1.499 rows=27 loops=1)
            -> Nested loop inner join  (cost=307.75 rows=63) (actual time=0.201..1.284 rows=135 loops=1)
                -> Filter: (i.`status` = 'available')  (cost=285.56 rows=63) (actual time=0.193..1.077 rows=135 loops=1)
                    -> Index range scan on i using PRICE, with index condition: (i.price < 5)  (cost=285.56 rows=634) (actual time=0.187..0.991 rows=634 loops=1)
                -> Single-row index lookup on c using PRIMARY (id=i.categoryId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=135)
```

Explanation: We find that the subquery relies heavily on selecting items with price less than 5, taking time 0.074..1.161, we decide to add an index to price. As a result, query time has been reduced.

CREATE INDEX STATUS
ON Item(Status)

(Revert:
ALTER TABLE Item
DROP INDEX STATUS)

```
-> Sort: Num_Item DESC  (actual time=1.167..1.173 rows=27 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.003 rows=27 loops=1)
        -> Aggregate using temporary table  (actual time=1.145..1.149 rows=27 loops=1)
            -> Nested loop inner join  (cost=107.18 rows=124) (actual time=0.205..0.953 rows=135 loops=1)
                -> Filter: (i.price < 5)  (cost=63.83 rows=124) (actual time=0.198..0.712 rows=135 loops=1)
                    -> Index lookup on i using STATUS (status='available')  (cost=63.83 rows=402) (actual time=0.195..0.680 rows=402 loops=1)
                -> Single-row index lookup on c using PRIMARY (id=i.categoryId)  (cost=0.25 rows=1) (actual time=0.002..0.002 rows=1 loops=135)
```

Explanation: We find that the subquery also relies on selecting items with available status, so we decide to add an index to status. As a result, query time has been reduced.

CREATE INDEX CATEGORY_ID
ON Item(categoryId)

(Revert:
ALTER TABLE Item
DROP INDEX CATEGORY_ID)

```
-> Sort: Num_Item DESC  (actual time=1.143..1.149 rows=27 loops=1)
    -> Table scan on <temporary>  (actual time=0.001..0.003 rows=27 loops=1)
        -> Aggregate using temporary table  (actual time=1.121..1.125 rows=27 loops=1)
            -> Nested loop inner join  (cost=107.18 rows=124) (actual time=0.207..0.903 rows=135 loops=1)
                -> Filter: (i.price < 5)  (cost=63.83 rows=124) (actual time=0.200..0.716 rows=135 loops=1)
                    -> Index lookup on i using STATUS (status='available')  (cost=63.83 rows=402) (actual time=0.197..0.685 rows=402 loops=1)
                -> Single-row index lookup on c using PRIMARY (id=i.categoryId)  (cost=0.25 rows=1) (actual time=0.001..0.001 rows=1 loops=135)
```

Explanation: Since category ID is used in the query, we tried to add an index to it. However, the query time has not been reduced. We presume that this is because the database compares a constant number of entries of category ID with IDs of category table in actual execution as shown in the analysis, so the index is not referenced at all.