

## Project Track 1, Stage 3

### Database Implementation and Indexing

#### **Database Implementation:**

Screenshot of our GCP connection to the database, and an example of 10 rows for the Medicines table.

Tables in stage2 schema			
carries			
causedBy			
currentlyHas			
medicines			
pharmacies			
side_effects			
symptoms			
treatedBy			
users			

9 rows in set (0.00 sec)

mysql> SELECT * FROM medicines LIMIT 10;			
name	manufacturer	composition	rating
A Doxid 100mg Capsule	Eskon Pharma	Doxycycline (100mg)	3.44
A Ret HC Cream	A. Menarini India Pvt Ltd	Hydroquinone (2% w/w) + Tretinoin (0.05% w/w) + Hydrocortisone (1% w/w)	2.44
A-CN Gel	Eskon Pharma	Clindamycin (1% w/w) + Nicotinamide (4% w/w)	3.66
A-Ret 0.025% Gel	A. Menarini India Pvt Ltd	Tretinoin (0.025% w/w)	3.00
A-Ret 0.05% Gel	A. Menarini India Pvt Ltd	Tretinoin (0.05% w/w)	3.06
A-Ret 0.1% Gel	A. Menarini India Pvt Ltd	Tretinoin (0.1% w/w)	3.00
A-Ret 0.5% Cream	A. Menarini India Pvt Ltd	Tretinoin (0.5% w/w)	3.16
AA 5 Tablet	Blaze Remedies	Levocetirizine (5mg)	5.00
AB Phylline Capsule	Sun Pharmaceutical Industries Ltd	Acebrophylline (100mg)	3.10
AB Phylline N Tablet	Sun Pharmaceutical Industries Ltd	Acebrophylline (100mg) + Acetylcysteine (600mg)	3.38

10 rows in set (0.00 sec)

#### **DDL commands:**

#### **FOR ENTITY TABLES:**

```
CREATE TABLE IF NOT EXISTS users (
state VARCHAR(255),
city VARCHAR(255),
address VARCHAR(255),
email VARCHAR(255) PRIMARY KEY,
preferred_pharmacy_id INT,
FOREIGN KEY (preferred_pharmacy_id) REFERENCES pharmacies(id) ON DELETE SET
NULL
);
```

```
CREATE TABLE IF NOT EXISTS symptoms (
name VARCHAR(255) PRIMARY KEY,
description VARCHAR(255)
);
```

```
CREATE TABLE IF NOT EXISTS medicines (
name VARCHAR(255) PRIMARY KEY,
manufacturer VARCHAR(255),
```

```
composition VARCHAR(255),
rating DECIMAL(3,2)
);
```

```
CREATE TABLE IF NOT EXISTS pharmacies (
state VARCHAR(255),
address VARCHAR(255),
city VARCHAR(255),
id INT AUTO_INCREMENT PRIMARY KEY
);
```

```
CREATE TABLE IF NOT EXISTS side_effects (
name VARCHAR(255) PRIMARY KEY,
description VARCHAR(255)
);
```

#### **FOR RELATIONSHIP TABLES:**

```
CREATE TABLE IF NOT EXISTS currentlyHas(
user_email VARCHAR(255),
symptom_name VARCHAR(255),
PRIMARY KEY (user_email, symptom_name),
FOREIGN KEY (symptom_name) REFERENCES symptoms(name) ON DELETE CASCADE,
FOREIGN KEY (user_email) REFERENCES users(email) ON DELETE CASCADE
);
```

```
CREATE TABLE IF NOT EXISTS treatedBy(
medicine_name VARCHAR(255),
symptom_name VARCHAR(255),
PRIMARY KEY (symptom_name, medicine_name),
FOREIGN KEY (symptom_name) REFERENCES symptoms(name) ON DELETE CASCADE,
FOREIGN KEY (medicine_name) REFERENCES medicines(name) ON DELETE CASCADE
);
```

```
CREATE TABLE IF NOT EXISTS causedBy(
medicine_name VARCHAR(255),
side_effect_name VARCHAR(255),
PRIMARY KEY (side_effect_name, medicine_name),
FOREIGN KEY (side_effect_name) REFERENCES side_effects(name) ON DELETE CASCADE,
FOREIGN KEY (medicine_name) REFERENCES medicines(name) ON DELETE CASCADE
);
```

```
CREATE TABLE IF NOT EXISTS carries(
```

```
medicine_name VARCHAR(255),
pharmacy_id INT,
PRIMARY KEY (pharmacy_id, medicine_name),
FOREIGN KEY (pharmacy_id) REFERENCES pharmacies(id) ON DELETE CASCADE,
FOREIGN KEY (medicine_name) REFERENCES medicines(name) ON DELETE CASCADE
);
```

#### **INSERT >=1000 ROWS INTO ALL TABLES:**

```
mysql> SELECT COUNT(*) FROM medicines;
+-----+
| COUNT(*) |
+-----+
|      11499 |
+-----+
1 row in set (0.01 sec)
```

```
mysql> SELECT COUNT(*) FROM pharmacies;
+-----+
| COUNT(*) |
+-----+
|      62973 |
+-----+
1 row in set (0.01 sec)
```

```
mysql> SELECT COUNT(*) FROM symptoms;
+-----+
| COUNT(*) |
+-----+
|      1005 |
+-----+
1 row in set (0.01 sec)
```

```
mysql> SELECT COUNT(*) FROM side_effects;
+-----+
| COUNT(*) |
+-----+
|      1160 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(*) FROM users;
+-----+
| COUNT(*) |
+-----+
|      2001 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(*) FROM currentlyHas;
+-----+
| COUNT(*) |
+-----+
|      4000 |
+-----+
1 row in set (0.00 sec)
```

```
mysql> SELECT COUNT(*) FROM treatedBy;
+-----+
| COUNT(*) |
+-----+
|      5020 |
+-----+
1 row in set (0.01 sec)
```

```
mysql> SELECT COUNT(*) FROM causedBy;
+-----+
| COUNT(*) |
+-----+
|     13795 |
+-----+
1 row in set (0.04 sec)
```

```
mysql> SELECT COUNT(*) FROM carries;
+-----+
| COUNT(*) |
+-----+
|    628207 |
+-----+
1 row in set (0.05 sec)
```

## ADVANCED QUERIES (below)

### 1) GET BEST RATED MEDICINES

The screenshot shows a MySQL query editor interface. The query is:

```
1 SELECT medicines.name AS medicine_name, COUNT(treatedBy.medicine_name) AS symptoms_treated, medicines.rating AS medicine_rating, medicines.manufacturer, medicines.composition
2 FROM stage2_schema.medicines AS medicines
3 JOIN stage2_schema.treatedBy AS treatedBy ON medicines.name = treatedBy.medicine_name
4 JOIN stage2_schema.currentlyHas AS currentlyHas ON treatedBy.symptom_name = currentlyHas.symptom_name
5 WHERE currentlyHas.user_email = 'a.brooks@outlook.com'
6 GROUP BY medicines.name
7 ORDER BY symptoms_treated DESC, medicine_rating DESC
8 LIMIT 15;
```

The results table has columns: medicine\_name, symptoms\_treated, medicine\_rating, manufacturer, and composition. The data is as follows:

medicine_name	symptoms_treated	medicine_rating	manufacturer	composition
Glifil M 2.5mg/250mg Tablet	1	5.00	Fourrts India Laboratories Pvt Ltd	Glibenclamide (2.5mg) + Metformin (250mg)
Oflo Ophthalmic Solution	1	5.00	Sunways India Pvt Ltd	Ofloxacin (0.3% w/v)
LCD 110 Tablet	1	4.12	Intas Pharmaceuticals Ltd	Levodopa (100mg) + Carbidopa (10mg)
Glimestar M 1 Tablet PR	1	3.58	Mankind Pharma Ltd	Glimepiride (1mg) + Metformin (500mg)
Kidpred Syrup	1	3.54	Abbott	Prednisolone (5mg/5ml)
Lyser D Tablet	1	3.28	Comed Chemicals Ltd	Diclofenac (50mg) + Serratiopeptidase (10mg)
Diapride M 0.5mg/500mg Tablet PR	1	3.22	Micro Labs Ltd	Glimepiride (0.5mg) + Metformin (500mg)
Mahaflox Eye Drop	1	2.86	Mankind Pharma Ltd	Moxifloxacin (0.5% w/v)
Terbest Cream	1	2.64	Systopic Laboratories Pvt Ltd	Terbinafine (1% w/w)
NE-C Tablet	1	1.44	Cachet Pharmaceuticals Pvt Ltd	Levo-carnitine (500mg) + Methylcobalamin (1500mcg) + Folic Acid (1.5mg)

Execution time: 1.5 ms

We think we get less than 15 results because of pure probability. We have 1005 symptoms and 11000 medicines. However, the symptoms that they treat are not optimally distributed, so it is not ~1000 medicines per symptom, but much less, especially taking into account that many user-reported symptoms are niche, such as "Abdominal Pain".

### 2) GET MEDICINES WITH LEAST SIDE EFFECTS

```

1 #medicines that treat a symptom with the least side effects
2 SELECT treatedBy.medicine_name AS medicines_name, COUNT(*) AS num_side_effects, medicines.rating, medicines.manufacturer, medicines.composition
3 FROM stage2_schema.medicines AS medicines
4 JOIN stage2_schema.causedBy AS causedBy ON medicines.name = causedBy.medicine_name
5 JOIN stage2_schema.treatedBy AS treatedBy ON causedBy.causedBy = treatedBy.causedBy
6 WHERE treatedBy.symptom_name = 'Abdominal pain'
7 GROUP BY treatedBy.medicine_name

```

**Results**

medicines_name	num_side_effects	rating	manufacturer	composition
Huminsulin R 100IU Cartridge	3	4.34	Eli Lilly and Company India Pvt Ltd	Human insulin (100IU)

Execution time: 1.7 ms [Export](#) [Copy](#)

Related to the reasoning for “get best rated medicine”, each integer band of side effects is broad. So there would only be a few medicines with the minimum number of side-effects. Here, Abdominal Pain only has around 5 medicines that treat it, and out of that, just having 1 with a minimum number makes sense. For symptoms such as cough, we would expect larger numbers, as many medicines treat it, and will tie for a minimum number of side effects. Even then, we expect not more than 8-10 tying medicines that all treat the same cure with the same minimum amount of side\_effects. We had some GCP credit and bucket issues - but will create larger datasets as we get that sorted for stage 4. However, this advanced query, as with all our others, works as expected.

### 3) GET THE PHARMACIES IN USER'S CITY/STATE WITH THE MOST MEDICINES

```

1 #pharmacies that carry the most medicines that can treat atleast one of your symptoms
2
3 SELECT pharmacies.id AS pharmacy_id, COUNT(DISTINCT carries.medicine_name) AS num_medicines_carried, pharmacies.address, pharmacies.city, pharmacies.state
4 FROM stage2_schema.pharmacies AS pharmacies
5 JOIN stage2_schema.carries AS carries ON pharmacies.id = carries.pharmacy_id
6 JOIN stage2_schema.treatedBy AS treatedBy ON carries.medicine_name = treatedBy.medicine_name
7 WHERE treatedBy.symptom_name = 'Abdominal pain'
8 GROUP BY pharmacy_id
9 LIMIT 15

```

**Results**

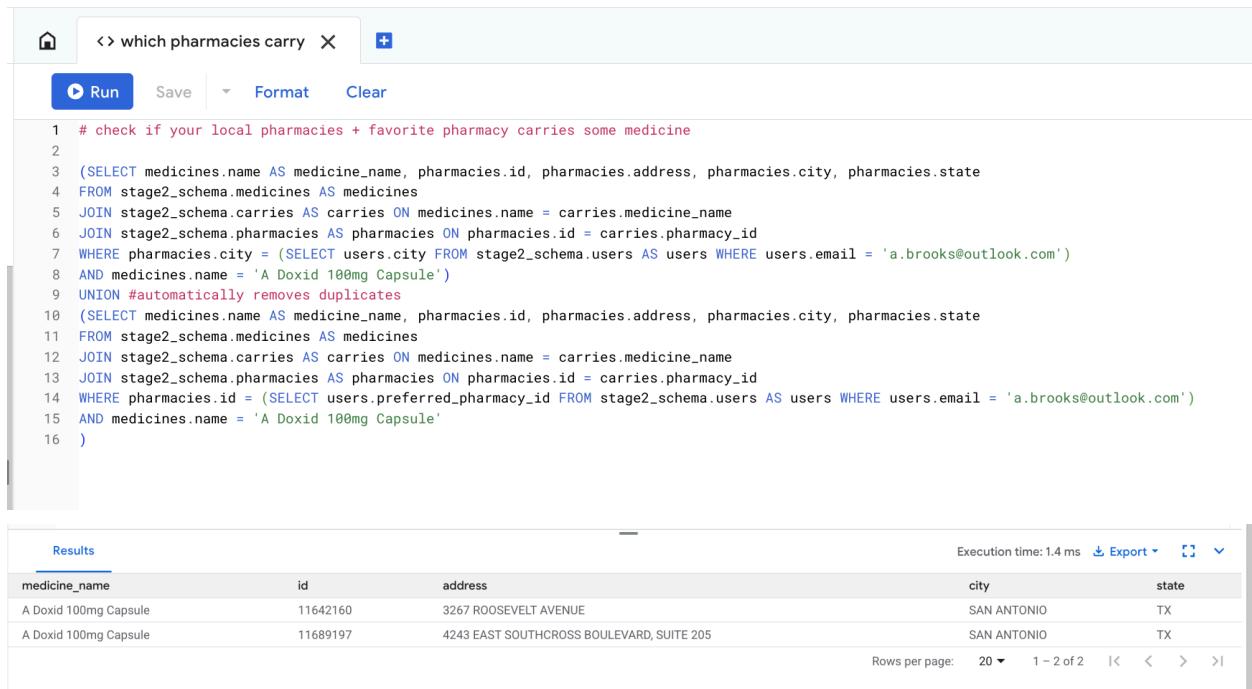
pharmacy_id	num_medicines_carried	address	city	state
11640640	1	1310 SOUTH 4TH STREET	NASHVILLE	AR
11641062	1	1915 WASHINGTON AVENUE	CAIRO	IL
11642236	1	207 GRAND STREET	NEW YORK	NY
11642311	1	940 TOWNE LAKE PARKWAY	WOODSTOCK	GA
11642401	1	76 GRAHAM AVENUE	BROOKLYN	NY
11642878	1	2211 SOUTH EOLA ROAD	AURORA	IL
11642939	1	421 SOUTH MAIN STREET	LEEDY	OK
11643035	1	1939 HOMER ROAD	COMMERCE	GA
11643042	1	1144 AIRPORT DRIVE	ALEXANDER CITY	AL
11643057	1	112 SOUTH MAIN STREET	SALISBURY	NC
11643169	1	2202 CHAPLINE STREET	WHEELING	WV
11644253	1	950 MALL LANE, BUILDING 950	TYNDALL AIR FORCE BASE	FL
11644339	1	331 SIJEN AVENUE	WHITEMAN AIR FORCE BASE	MO
11644991	1	1855 ALUM ROCK AVENUE, SUITE A	SAN JOSE	CA

Execution time: 2.2 ms [Export](#) [Copy](#)

Notice that each resulting query has only 1 medicine that treats the user. Each Pharmacy only carries a small set {5,15} medicines, in our synthetic data. Given that are ~63,000 pharmacies that carry a random sample of {5,15} out of ~11,500 medicines, the

probability that a user's symptom is treated by pharmacies in a given state {on average, a state seems to have 500 pharmacies}, is low. However, in stage 4, we will synthesize a larger number of medicines per pharmacy, and this number should be much larger. However, even with the current data, we have 274 pharmacies this user can go to in their US state/territory. So we expect to get 15+ for this, unless the state/territory itself has less than 100 pharmacies.

#### 4) GET THE PHARMACIES THAT CARRY MEDICINES TO CURE USER



The screenshot shows a database query interface with the following details:

**Query:**

```

1 # check if your local pharmacies + favorite pharmacy carries some medicine
2
3 (SELECT medicines.name AS medicine_name, pharmacies.id, pharmacies.address, pharmacies.city, pharmacies.state
4 FROM stage2_schema.medicines AS medicines
5 JOIN stage2_schema.carries AS carries ON medicines.name = carries.medicine_name
6 JOIN stage2_schema.pharmacies AS pharmacies ON pharmacies.id = carries.pharmacy_id
7 WHERE pharmacies.city = (SELECT users.city FROM stage2_schema.users AS users WHERE users.email = 'a.brooks@outlook.com')
8 AND medicines.name = 'A Doxid 100mg Capsule'
9 UNION #automatically removes duplicates
10 (SELECT medicines.name AS medicine_name, pharmacies.id, pharmacies.address, pharmacies.city, pharmacies.state
11 FROM stage2_schema.medicines AS medicines
12 JOIN stage2_schema.carries AS carries ON medicines.name = carries.medicine_name
13 JOIN stage2_schema.pharmacies AS pharmacies ON pharmacies.id = carries.pharmacy_id
14 WHERE pharmacies.id = (SELECT users.preferred_pharmacy_id FROM stage2_schema.users AS users WHERE users.email = 'a.brooks@outlook.com')
15 AND medicines.name = 'A Doxid 100mg Capsule'
16 )

```

**Results:**

medicine_name	id	address	city	state
A Doxid 100mg Capsule	11642160	3267 ROOSEVELT AVENUE	SAN ANTONIO	TX
A Doxid 100mg Capsule	11689197	4243 EAST SOUTHCROSS BOULEVARD, SUITE 205	SAN ANTONIO	TX

Execution time: 1.4 ms    Export ▾

This is the most specific of our queries, and has a lot of specific predicates to check whether a certain favorite pharmacy of the current user carries medicine that treats the current user's current symptom. We expected this to be the least, but as we will synthesize extensions for a lot of our datasets, we expect to see a few more queries. However, we think that 15+ will still be unlikely for this one.

## INDEX ANALYSIS:

### Pharmacy Most Medicine Default:

The screenshot shows a database query interface with the following details:

- Toolbar:** Run, Save, Format, Clear, Syntax error at or near "ANALYZE"
- Query:**

```

1 #pharmacies that carry the most medicines that can treat atleast one of your symptoms
2 EXPLAIN ANALYZE
3 SELECT pharmacies.id AS pharmacy_id, COUNT(DISTINCT carries.medicine_name) AS num_medicines_carried, pharmacies.state, pharmacies.
address, pharmacies.city
4 FROM stage2_schema.pharmacies AS pharmacies
5 JOIN stage2_schema.carries AS carries ON pharmacies.id = carries.pharmacy_id
6 JOIN stage2_schema.treatedBy AS treatedBy ON carries.medicine_name = treatedBy.medicine_name
7 WHERE treatedBy.symptom_name = 'Abdominal pain'
8 GROUP BY pharmacy_id
9 LIMIT 15;

```
- Results Tab:** Execution time: 1.9 ms, Export, Rows per page: 20, 1 - 1 of 1
- EXPLAIN Output:**

```

-> Limit: 15 row(s) (actual time=0.814..0.82 rows=15 loops=1) -> Group aggregate: count(distinct carries.medicine_name) (actual time=0.814..0.819 rows=15
loops=1) -> Sort: carries.pharmacy_id (actual time=0.808..0.81 rows=16 loops=1) -> Stream results (cost=197 rows=273) (actual time=0.0485..0.737 rows=274
loops=1) -> Nested loop inner join (cost=197 rows=273) (actual time=0.0469..0.638 rows=274 loops=1) -> Nested loop inner join (cost=101 rows=273) (actual
time=0.0398..0.202 rows=274 loops=1) -> Index lookup on treatedBy using symptom_name (symptom_name='Abdominal pain') (cost=1.75 rows=5) (actual
time=0.0245..0.0303 rows=5 loops=1) -> Covering index lookup on carries using idx_med_pharm (medicine_name=treatedBy.medicine_name) (cost=15.5
rows=54.7) (actual time=0.00852..0.0317 rows=54.8 loops=5) -> Single-row index lookup on pharmacies using PRIMARY (id=carries.pharmacy_id) (cost=0.25
rows=1) (actual time=0.00145..0.00147 rows=1 loops=274)

```

### Pharmacy Most Medicine + Pharmacy.Address:

The screenshot shows a database query interface with the following details:

- Toolbar:** Run, Save, Format, Clear, Syntax error at or near "ANALYZE"
- Query:**

```

1 #pharmacies that carry the most medicines that can treat atleast one of your symptoms
2 EXPLAIN ANALYZE
3 SELECT pharmacies.id AS pharmacy_id, COUNT(DISTINCT carries.medicine_name) AS num_medicines_carried, pharmacies.state, pharmacies.
address, pharmacies.city
4 FROM stage2_schema.pharmacies AS pharmacies
5 JOIN stage2_schema.carries AS carries ON pharmacies.id = carries.pharmacy_id
6 JOIN stage2_schema.treatedBy AS treatedBy ON carries.medicine_name = treatedBy.medicine_name
7 WHERE treatedBy.symptom_name = 'Abdominal pain'
8 GROUP BY pharmacy_id
9 LIMIT 15;

```
- Results Tab:** Execution time: 1.3 ms, Execution time: 7.0 ms, Export, Rows per page: 20, 1 - 1 of 1
- EXPLAIN Output:**

```

-> Limit: 15 row(s) (actual time=1.3..1.31 rows=15 loops=1) -> Group aggregate: count(distinct carries.medicine_name) (actual time=1.29..1.3 rows=15 loops=1) ->
Sort: carries.pharmacy_id (actual time=1.29..1.29 rows=16 loops=1) -> Stream results (cost=197 rows=273) (actual time=0.0472..1.17 rows=274 loops=1) ->
Nested loop inner join (cost=197 rows=273) (actual time=0.0456..0.994 rows=274 loops=1) -> Nested loop inner join (cost=101 rows=273) (actual
time=0.0382..0.36 rows=274 loops=1) -> Index lookup on treatedBy using symptom_name (symptom_name='Abdominal pain') (cost=1.75 rows=5) (actual
time=0.0179..0.029 rows=5 loops=1) -> Covering index lookup on carries using idx_med_pharm (medicine_name=treatedBy.medicine_name) (cost=15.5
rows=54.7) (actual time=0.0146..0.0619 rows=54.8 loops=5) -> Single-row index lookup on pharmacies using PRIMARY (id=carries.pharmacy_id) (cost=0.25
rows=1) (actual time=0.00209..0.00213 rows=1 loops=274)

```

### Pharmacy Most Medicine + Users.city + Pharmacies.city

Run Save Format Clear Syntax error at or near "ANALYZE"

```

1 #pharmacies that carry the most medicines that can treat atleast one of your symptoms
2 EXPLAIN ANALYZE
3 SELECT pharmacies.id AS pharmacy_id, COUNT(DISTINCT carries.medicine_name) AS num_medicines_carried, pharmacies.state, pharmacies.
address, pharmacies.city
4 FROM stage2_schema.pharmacies AS pharmacies
5 JOIN stage2_schema.carries AS carries ON pharmacies.id = carries.pharmacy_id
6 JOIN stage2_schema.treatedBy AS treatedBy ON carries.medicine_name = treatedBy.medicine_name
7 WHERE treatedBy.symptom_name = 'Abdominal pain'
8 GROUP BY pharmacy_id
9 LIMIT 15;

```

Results Execution time: 1.3 ms Execution time: 1.9 ms Export ▾

**EXPLAIN**

```

-> Limit: 15 row(s) (actual time=0.827..0.833 rows=15 loops=1) -> Group aggregate: count(distinct carries.medicine_name) (actual time=0.826..0.832
rows=15 loops=1) -> Sort: carries.pharmacy_id (actual time=0.821..0.822 rows=16 loops=1) -> Stream results (cost=197 rows=273) (actual
time=0.0565..0.755 rows=274 loops=1) -> Nested loop inner join (cost=197 rows=273) (actual time=0.0549..0.652 rows=274 loops=1) -> Nested loop inner
join (cost=101 rows=273) (actual time=0.0468..0.221 rows=274 loops=1) -> Index lookup on treatedBy using symptom_name (symptom_name='Abdominal
pain') (cost=1.75 rows=5) (actual time=0.0255..0.0317 rows=5 loops=1) -> Covering index lookup on carries using idx_med_pharm
(medicine_name=treatedBy.medicine_name) (cost=15.5 rows=54.7) (actual time=0.00999..0.0351 rows=54.8 loops=5) -> Single-row index lookup on
pharmacies using PRIMARY (id=carries.pharmacy_id) (cost=0.25 rows=1) (actual time=0.00143..0.00145 rows=1 loops=274)

```

Rows per page: 20 ▾ 1 – 1 of 1 | < > >|

## Pharmacy Most Medicine + Users.state + Pharmacies.state

Run Save Format Clear Syntax error at or near "ANALYZE"

```

1 #pharmacies that carry the most medicines that can treat atleast one of your symptoms
2 EXPLAIN ANALYZE
3 SELECT pharmacies.id AS pharmacy_id, COUNT(DISTINCT carries.medicine_name) AS num_medicines_carried, pharmacies.state, pharmacies.
address, pharmacies.city
4 FROM stage2_schema.pharmacies AS pharmacies
5 JOIN stage2_schema.carries AS carries ON pharmacies.id = carries.pharmacy_id
6 JOIN stage2_schema.treatedBy AS treatedBy ON carries.medicine_name = treatedBy.medicine_name
7 WHERE treatedBy.symptom_name = 'Abdominal pain'
8 GROUP BY pharmacy_id
9 LIMIT 15;

```

Results Execution time: 1.3 ms Execution time: 6.0 ms Export ▾

**EXPLAIN**

```

-> Limit: 15 row(s) (actual time=2.16..2.17 rows=15 loops=1) -> Group aggregate: count(distinct carries.medicine_name) (actual time=2.16..2.17 rows=15
loops=1) -> Sort: carries.pharmacy_id (actual time=2.15..2.15 rows=16 loops=1) -> Stream results (cost=197 rows=273) (actual time=0.0639..2.02 rows=274
loops=1) -> Nested loop inner join (cost=197 rows=273) (actual time=0.0622..1.05 rows=274 loops=1) -> Nested loop inner join (cost=101 rows=273) (actual
time=0.0527..0.389 rows=274 loops=1) -> Index lookup on treatedBy using symptom_name (symptom_name='Abdominal pain') (cost=1.75 rows=5) (actual
time=0.0294..0.0433 rows=5 loops=1) -> Covering index lookup on carries using idx_med_pharm (medicine_name=treatedBy.medicine_name) (cost=15.5
rows=54.7) (actual time=0.0156..0.0649 rows=54.8 loops=5) -> Single-row index lookup on pharmacies using PRIMARY (id=carries.pharmacy_id) (cost=0.25
rows=1) (actual time=0.0022..0.00223 rows=1 loops=274)

```

Rows per page: 20 ▾ 1 – 1 of 1 | < > >|

For the pharmacy most medicine query, the default cost is 197. We experimented with indexes on `pharmacy.address`, `users.city + pharmacies.city`, and `users.state + pharmacies.state`. None of these indexing options reduced the query cost. This suggests that the query planner already chooses an optimal access path—likely dominated by join operations or aggregates where filtering by address, city, or state contributes little selectivity. Therefore, the default indexing strategy remains optimal.

Get Best Medicine Default:

Run Save Format Clear Syntax error at or near "ANALYZE"

```

1 EXPLAIN ANALYZE
2 SELECT medicines.name AS medicine_name, COUNT(treatedBy.medicine_name) AS symptoms_treated, medicines.rating AS medicine_rating,
medicines.manufacturer, medicines.composition
3 FROM stage2_schema.medicines AS medicines
4 JOIN stage2_schema.treatedBy AS treatedBy ON medicines.name = treatedBy.medicine_name
5 JOIN stage2_schema.currentlyHas AS currentlyHas ON treatedBy.symptom_name = currentlyHas.symptom_name
6 WHERE currentlyHas.user_email = 'a.brooks@outlook.com'
7 GROUP BY medicines.name
8 ORDER BY symptoms_treated DESC, medicine_rating DESC
9 LIMIT 15;

```

Results Execution time: 4.2 ms Export ▾

**EXPLAIN**

```

-> Limit: 15 row(s) (actual time=0.168..0.17 rows=10 loops=1) -> Sort: symptoms_treated DESC, medicines.rating DESC, limit input to 15 row(s) per chunk (actual time=0.168..0.169 rows=10 loops=1) -> Table scan on <temporary> (actual time=0.149..0.151 rows=10 loops=1) -> Aggregate using temporary table (actual time=0.147..0.147 rows=10 loops=1) -> Nested loop inner join (cost=7.91 rows=10.3) (actual time=0.0368..0.111 rows=10 loops=1) -> Nested loop inner join (cost=4.31 rows=10.3) (actual time=0.0282..0.0624 rows=10 loops=1) -> Index lookup on currentlyHas using user_email (user_email='a.brooks@outlook.com') (cost=0.7 rows=2) (actual time=0.0142..0.0182 rows=2 loops=1) -> Index lookup on treatedBy using symptom_name (symptom_name=currentlyHas.symptom_name) (cost=1.55 rows=5.15) (actual time=0.011..0.0212 rows=5 loops=2) -> Single-row index lookup on medicines using PRIMARY (name=treatedBy.medicine_name) (cost=0.26 rows=1) (actual time=0.00389..0.00392 rows=1 loops=10)

```

Rows per page: 20 ▾ 1 – 1 of 1 | < > >>

## Get Best Medicine + Medicine.ratings:

Run Save Format Clear Syntax error at or near "ANALYZE"

```

1 EXPLAIN ANALYZE
2 SELECT medicines.name AS medicine_name, COUNT(treatedBy.medicine_name) AS symptoms_treated, medicines.rating AS medicine_rating,
medicines.manufacturer, medicines.composition
3 FROM stage2_schema.medicines AS medicines
4 JOIN stage2_schema.treatedBy AS treatedBy ON medicines.name = treatedBy.medicine_name
5 JOIN stage2_schema.currentlyHas AS currentlyHas ON treatedBy.symptom_name = currentlyHas.symptom_name
6 WHERE currentlyHas.user_email = 'a.brooks@outlook.com'
7 GROUP BY medicines.name
8 ORDER BY symptoms_treated DESC, medicine_rating DESC
9 LIMIT 15;

```

Results Execution time: 1.3 ms Execution time: 2.4 ms Export ▾

**EXPLAIN**

```

-> Limit: 15 row(s) (actual time=0.15..0.159 rows=10 loops=1) -> Sort: symptoms_treated DESC, medicines.rating DESC, limit input to 15 row(s) per chunk (actual time=0.15..0.157 rows=10 loops=1) -> Table scan on <temporary> (actual time=0.131..0.134 rows=10 loops=1) -> Aggregate using temporary table (actual time=0.13..0.13 rows=10 loops=1) -> Nested loop inner join (cost=7.91 rows=10.3) (actual time=0.031..0.092 rows=10 loops=1) -> Nested loop inner join (cost=4.31 rows=10.3) (actual time=0.0241..0.0544 rows=10 loops=1) -> Index lookup on currentlyHas using user_email (user_email='a.brooks@outlook.com') (cost=0.7 rows=2) (actual time=0.0129..0.0163 rows=2 loops=1) -> Index lookup on treatedBy using symptom_name (symptom_name=currentlyHas.symptom_name) (cost=1.55 rows=5.15) (actual time=0.00943..0.0182 rows=5 loops=2) -> Single-row index lookup on medicines using PRIMARY (name=treatedBy.medicine_name) (cost=0.26 rows=1) (actual time=0.00344..0.00347 rows=1 loops=10)

```

Rows per page: 20 ▾ 1 – 1 of 1 | < > >>

## Get Best Medicine + Medicine.manufacturer:

Run Save Format Clear Syntax error at or near "ANALYZE"

```

1 EXPLAIN ANALYZE
2 SELECT medicines.name AS medicine_name, COUNT(treatedBy.medicine_name) AS symptoms_treated, medicines.rating AS medicine_rating,
   medicines.manufacturer, medicines.composition
3 FROM stage2_schema.medicines AS medicines
4 JOIN stage2_schema.treatedBy AS treatedBy ON medicines.name = treatedBy.medicine_name
5 JOIN stage2_schema.currentlyHas AS currentlyHas ON treatedBy.symptom_name = currentlyHas.symptom_name
6 WHERE currentlyHas.user_email = 'a.brooks@outlook.com'
7 GROUP BY medicines.name
8 ORDER BY symptoms_treated DESC, medicine_rating DESC
9 LIMIT 15;

```

Results Execution time: 1.3 ms Execution time: 1.6 ms Export ▾

**EXPLAIN**

```

-> Limit: 15 row(s) (actual time=0.111..0.111 rows=10 loops=1) -> Sort: symptoms_treated DESC, medicines.rating DESC, limit input to 15 row(s) per chunk
(actual time=0.109..0.111 rows=10 loops=1) -> Table scan on <temporary> (actual time=0.0958..0.0981 rows=10 loops=1) -> Aggregate using temporary table
(actual time=0.0947..0.0947 rows=10 loops=1) -> Nested loop inner join (cost=7.91 rows=10.3) (actual time=0.0249..0.0695 rows=10 loops=1) -> Nested
loop inner join (cost=4.31 rows=10.3) (actual time=0.0189..0.0405 rows=10 loops=1) -> Index lookup on currentlyHas using user_email
(user_email='a.brooks@outlook.com') (cost=0.7 rows=2) (actual time=0.00937..0.0116 rows=2 loops=1) -> Index lookup on treatedBy using symptom_name
(symptom_name=currentlyHas.symptom_name) (cost=1.55 rows=5.15) (actual time=0.00728..0.0137 rows=5 loops=2) -> Single-row index lookup on
medicines using PRIMARY (name=treatedBy.medicine_name) (cost=0.26 rows=1) (actual time=0.00264..0.00266 rows=1 loops=10)

```

Rows per page: 20 ▾ 1 – 1 of 1 | < < > > |

## Get Best Medicine + Medicine.composition:

Run Save Format Clear Syntax error at or near "ANALYZE"

```

1 EXPLAIN ANALYZE
2 SELECT medicines.name AS medicine_name, COUNT(treatedBy.medicine_name) AS symptoms_treated, medicines.rating AS medicine_rating,
   medicines.manufacturer, medicines.composition
3 FROM stage2_schema.medicines AS medicines
4 JOIN stage2_schema.treatedBy AS treatedBy ON medicines.name = treatedBy.medicine_name
5 JOIN stage2_schema.currentlyHas AS currentlyHas ON treatedBy.symptom_name = currentlyHas.symptom_name
6 WHERE currentlyHas.user_email = 'a.brooks@outlook.com'
7 GROUP BY medicines.name
8 ORDER BY symptoms_treated DESC, medicine_rating DESC
9 LIMIT 15;

```

Results Execution time: 1.3 ms Execution time: 1.8 ms Export ▾

**EXPLAIN**

```

-> Limit: 15 row(s) (actual time=0.151..0.153 rows=10 loops=1) -> Sort: symptoms_treated DESC, medicines.rating DESC, limit input to 15 row(s) per chunk
(actual time=0.151..0.152 rows=10 loops=1) -> Table scan on <temporary> (actual time=0.132..0.134 rows=10 loops=1) -> Aggregate using temporary table
(actual time=0.13..0.13 rows=10 loops=1) -> Nested loop inner join (cost=7.91 rows=10.3) (actual time=0.0315..0.0912 rows=10 loops=1) -> Nested loop
inner join (cost=4.31 rows=10.3) (actual time=0.0227..0.0534 rows=10 loops=1) -> Index lookup on currentlyHas using user_email
(user_email='a.brooks@outlook.com') (cost=0.7 rows=2) (actual time=0.0116..0.0152 rows=2 loops=1) -> Index lookup on treatedBy using symptom_name
(symptom_name=currentlyHas.symptom_name) (cost=1.55 rows=5.15) (actual time=0.00917..0.0182 rows=5 loops=2) -> Single-row index lookup on
medicines using PRIMARY (name=treatedBy.medicine_name) (cost=0.26 rows=1) (actual time=0.00346..0.00349 rows=1 loops=10)

```

Rows per page: 20 ▾ 1 – 1 of 1 | < < > > |

For the get best medicine query, the default cost is 7.91. We tested additional indexes on medicine.ratings, medicine.manufacturer, and medicine.composition. Since all versions retained the same estimated cost, these attributes do not constrain the query enough to make index lookups more efficient than the existing plan. Hence, the default indexing is already optimal for this query.

## Medicine Least Side Effects Default:

Run Save Format Clear Syntax error at or near "ANALYZE"

```

1 #medicines that treat a symptom with the least side effects
2 EXPLAIN ANALYZE
3 SELECT treatedBy.medicine_name AS medicines_name, COUNT(*) AS num_side_effects, medicines.manufacturer, medicines.composition,
       medicines.rating
4 FROM stage2_schema.medicines AS medicines
5 JOIN stage2_schema.causedBy AS causedBy ON medicines.name = causedBy.medicine_name
6 JOIN stage2_schema.treatedBy AS treatedBy ON causedBy.causedBy_name = treatedBy.medicine_name
7 WHERE treatedBy.symptom_name = 'Abdominal pain'
8 GROUP BY treatedBy.medicine_name
9 LIMIT 15;

```

Results Execution time: 1.1 ms Export ▾

EXPLAIN

```

-> Limit: 15 row(s) (actual time=0.136..0.137 rows=1 loops=1) -> Table scan on <temporary> (actual time=0.135..0.136 rows=1 loops=1) -> Aggregate using temporary table (actual time=0.134..0.134 rows=1 loops=1) -> Nested loop inner join (cost=11.1 rows=10.7) (actual time=0.0695..0.105 rows=3 loops=1) -> Nested loop inner join (cost=3.5 rows=5) (actual time=0.0403..0.0683 rows=5 loops=1) -> Index lookup on treatedBy using symptom_name (symptom_name='Abdominal pain') (cost=1.75 rows=5) (actual time=0.0285..0.0389 rows=5 loops=1) -> Single-row index lookup on medicines using PRIMARY (name=treatedBy.medicine_name) (cost=0.27 rows=1) (actual time=0.00537..0.0054 rows=1 loops=5) -> Covering index lookup on causedBy using PRIMARY (medicine_name=treatedBy.medicine_name) (cost=1.35 rows=2.14) (actual time=0.00651..0.00686 rows=0.6 loops=5)

```

Rows per page: 20 ▾ 1 – 1 of 1 | < < > >|

## Medicine Least Side Effects + Medicine.ratings:

Run Save Format Clear Syntax error at or near "ANALYZE"

```

1 #medicines that treat a symptom with the least side effects
2 EXPLAIN ANALYZE
3 SELECT treatedBy.medicine_name AS medicines_name, COUNT(*) AS num_side_effects, medicines.manufacturer, medicines.composition,
       medicines.rating
4 FROM stage2_schema.medicines AS medicines
5 JOIN stage2_schema.causedBy AS causedBy ON medicines.name = causedBy.medicine_name
6 JOIN stage2_schema.treatedBy AS treatedBy ON causedBy.causedBy_name = treatedBy.medicine_name
7 WHERE treatedBy.symptom_name = 'Abdominal pain'
8 GROUP BY treatedBy.medicine_name
9 LIMIT 15;

```

Results Execution time: 1.3 ms Execution time: 1.6 ms Export ▾

EXPLAIN

```

-> Limit: 15 row(s) (actual time=0.114..0.114 rows=1 loops=1) -> Table scan on <temporary> (actual time=0.113..0.113 rows=1 loops=1) -> Aggregate using temporary table (actual time=0.112..0.112 rows=1 loops=1) -> Nested loop inner join (cost=11.1 rows=10.7) (actual time=0.0512..0.088 rows=3 loops=1) -> Nested loop inner join (cost=3.5 rows=5) (actual time=0.0254..0.0529 rows=5 loops=1) -> Index lookup on treatedBy using symptom_name (symptom_name='Abdominal pain') (cost=1.75 rows=5) (actual time=0.0155..0.025 rows=5 loops=1) -> Single-row index lookup on medicines using PRIMARY (name=treatedBy.medicine_name) (cost=0.27 rows=1) (actual time=0.00509..0.00513 rows=1 loops=5) -> Covering index lookup on causedBy using PRIMARY (medicine_name=treatedBy.medicine_name) (cost=1.35 rows=2.14) (actual time=0.00622..0.00659 rows=0.6 loops=5)

```

Rows per page: 20 ▾ 1 – 1 of 1 | < < > >|

## Medicine Least Side Effects + Medicine.composition:

Run Save Format Clear Syntax error at or near "ANALYZE"

```

1 #medicines that treat a symptom with the least side effects
2 EXPLAIN ANALYZE
3 SELECT treatedBy.medicine_name AS medicines_name, COUNT(*) AS num_side_effects, medicines.manufacturer, medicines.composition,
medicines.rating
4 FROM stage2_schema.medicines AS medicines
5 JOIN stage2_schema.causedBy AS causedBy ON medicines.name = causedBy.medicine_name
6 JOIN stage2_schema.treatedBy AS treatedBy ON causedBy.causedBy_name = treatedBy.medicine_name
7 WHERE treatedBy.symptom_name = 'Abdominal pain'
8 GROUP BY treatedBy.medicine_name
9 LIMIT 15;

```

Results Execution time: 1.3 ms Execution time: 1.3 ms Export ▾

**EXPLAIN**

-> Limit: 15 row(s) (actual time=0.123..0.123 rows=1 loops=1) -> Table scan on <temporary> (actual time=0.122..0.122 rows=1 loops=1) -> Aggregate using temporary table (actual time=0.121..0.121 rows=1 loops=1) -> Nested loop inner join (cost=11.1 rows=10.7) (actual time=0.0591..0.0959 rows=3 loops=1) -> Nested loop inner join (cost=3.5 rows=5) (actual time=0.0344..0.0619 rows=5 loops=1) -> Index lookup on treatedBy using symptom\_name (symptom\_name='Abdominal pain') (cost=1.75 rows=5) (actual time=0.0232..0.0335 rows=5 loops=1) -> Single-row index lookup on medicines using PRIMARY (name=treatedBy.medicine\_name) (cost=0.27 rows=1) (actual time=0.00522..0.00526 rows=1 loops=5) -> Covering index lookup on causedBy using PRIMARY (medicine\_name=treatedBy.medicine\_name) (cost=1.35 rows=2.14) (actual time=0.006..0.00637 rows=0.6 loops=5)

Rows per page: 20 ▾ 1 – 1 of 1 |< < > >|

## Medicine Least Side Effects + Medicine.manufacturer:

Run Save Format Clear Syntax error at or near "ANALYZE"

```

1 #medicines that treat a symptom with the least side effects
2 EXPLAIN ANALYZE
3 SELECT treatedBy.medicine_name AS medicines_name, COUNT(*) AS num_side_effects, medicines.manufacturer, medicines.composition,
medicines.rating
4 FROM stage2_schema.medicines AS medicines
5 JOIN stage2_schema.causedBy AS causedBy ON medicines.name = causedBy.medicine_name
6 JOIN stage2_schema.treatedBy AS treatedBy ON causedBy.causedBy_name = treatedBy.medicine_name
7 WHERE treatedBy.symptom_name = 'Abdominal pain'
8 GROUP BY treatedBy.medicine_name
9 LIMIT 15; |

```

Results Execution time: 1.3 ms Execution time: 1.2 ms Export ▾

**EXPLAIN**

-> Limit: 15 row(s) (actual time=0.133..0.134 rows=1 loops=1) -> Table scan on <temporary> (actual time=0.132..0.133 rows=1 loops=1) -> Aggregate using temporary table (actual time=0.131..0.131 rows=1 loops=1) -> Nested loop inner join (cost=11.1 rows=10.7) (actual time=0.0605..0.101 rows=3 loops=1) -> Nested loop inner join (cost=3.5 rows=5) (actual time=0.0318..0.0649 rows=5 loops=1) -> Index lookup on treatedBy using symptom\_name (symptom\_name='Abdominal pain') (cost=1.75 rows=5) (actual time=0.0208..0.0313 rows=5 loops=1) -> Single-row index lookup on medicines using PRIMARY (name=treatedBy.medicine\_name) (cost=0.27 rows=1) (actual time=0.00626..0.00629 rows=1 loops=5) -> Covering index lookup on causedBy using PRIMARY (medicine\_name=treatedBy.medicine\_name) (cost=1.35 rows=2.14) (actual time=0.0064..0.00677 rows=0.6 loops=5)

Rows per page: 20 ▾ 1 – 1 of 1 |< < > >|

For the medicine least side effects query, the default cost is 11.1. We tried indexing medicine.ratings, medicine.composition and medicine.manufacturer. Neither option improved the cost, indicating that the query's filtering or ordering criteria are already efficiently handled

without additional indexes. The query optimizer continues to prefer sequential or existing indexed access paths, so the default indexing remains optimal.

### Which Pharmacies Carry Default:

The screenshot shows a database query interface with the following details:

- Query:**

```
1 # check if your local pharmacies + favorite pharmacy carries some medicine
2 EXPLAIN ANALYZE
3 (SELECT medicines.name AS medicine_name, pharmacies.id, pharmacies.state, pharmacies.address, pharmacies.city
4 FROM stage2_schema.medicines AS medicines
5 JOIN stage2_schema.carries AS carries ON medicines.name = carries.medicine_name
6 JOIN stage2_schema.pharmacies AS pharmacies ON pharmacies.id = carries.pharmacy_id
7 WHERE pharmacies.city = (SELECT users.city FROM stage2_schema.users AS users WHERE users.email = 'a.brooks@outlook.com')
8 AND medicines.name = 'A Doxid 100mg Capsule'
9 UNION #automatically removes duplicates
10 (SELECT medicines.name AS medicine_name, pharmacies.id, pharmacies.state, pharmacies.address, pharmacies.city
11 FROM stage2_schema.medicines AS medicines
12 JOIN stage2_schema.carries AS carries ON medicines.name = carries.medicine_name
13 JOIN stage2_schema.pharmacies AS pharmacies ON pharmacies.id = carries.pharmacy_id
```
- EXPLAIN Output:**

```
-> Table scan on <union temporary> (cost=38.2..40.2 rows=5.2) (actual time=0.169..0.17 rows=2 loops=1) -> Union materialize with deduplication
(cost=37.7..37.7 rows=5.2) (actual time=0.167..0.167 rows=2 loops=1) -> Nested loop inner join (cost=37.1 rows=5.2) (actual time=0.0167..0.156 rows=2
loops=1) -> Covering index lookup on carries using idx_med_pharm (medicine_name='A Doxid 100mg Capsule') (cost=18.9 rows=52) (actual time=0.0082..0.0344
rows=52 loops=1) -> Filter: (pharmacies.city = (select #2)) (cost=0.25 rows=0.1) (actual time=0.00223..0.00223 rows=0.0385 loops=52) -> Single-row index
lookup on pharmacies using PRIMARY (id=carries.pharmacy_id) (cost=0.25 rows=1) (actual time=0.00171..0.00175 rows=1 loops=52) -> Select #2 (subquery in
condition; run only once) -> Rows fetched before execution (cost=0.0 rows=1) (actual time=58e-6..103e-6 rows=1 loops=1) -> Zero rows (no matching row in
const table) (cost=0..0 rows=0) (actual time=53e-6..53e-6 rows=0 loops=1)
```
- Execution Time:** 1.3 ms
- Results:** (No results shown)

### Which Pharmacies Carry + Pharmacy.address:

Run Save Format Clear Syntax error at or near "ANALYZE"

```

1 # check if your local pharmacies + favorite pharmacy carries some medicine
2 EXPLAIN ANALYZE
3 (SELECT medicines.name AS medicine_name, pharmacies.id, pharmacies.state, pharmacies.address, pharmacies.city
4 FROM stage2_schema.medicines AS medicines
5 JOIN stage2_schema.carries AS carries ON medicines.name = carries.medicine_name
6 JOIN stage2_schema.pharmacies AS pharmacies ON pharmacies.id = carries.pharmacy_id
7 WHERE pharmacies.city = (SELECT users.city FROM stage2_schema.users AS users WHERE users.email = 'a.brooks@outlook.com')
8 AND medicines.name = 'A Doxid 100mg Capsule'
9 UNION #automatically removes duplicates
10 (SELECT medicines.name AS medicine_name, pharmacies.id, pharmacies.state, pharmacies.address, pharmacies.city
11 FROM stage2_schema.medicines AS medicines
12 JOIN stage2_schema.carries AS carries ON medicines.name = carries.medicine_name
13 JOIN stage2_schema.pharmacies AS pharmacies ON pharmacies.id = carries.pharmacy_id

```

Results Execution time: 1.3 ms Execution time: 2.3 ms Export ▾

**EXPLAIN**

-> Table scan on <union temporary> (cost=38.2..40.2 rows=5.2) (actual time=0.21..0.21 rows=2 loops=1) -> Union materialize with deduplication (cost=37.7..37.7 rows=5.2) (actual time=0.208..0.208 rows=2 loops=1) -> Nested loop inner join (cost=37.1 rows=5.2) (actual time=0.0246..0.195 rows=2 loops=1) -> Covering index lookup on carries using idx\_med\_pharm (medicine\_name='A Doxid 100mg Capsule') (cost=18.9 rows=52) (actual time=0.0126..0.0515 rows=52 loops=1) -> Filter: (pharmacies.city = (select #2)) (cost=0.25 rows=0.1) (actual time=0.0026..0.00261 rows=0.0385 loops=52) -> Single-row index lookup on pharmacies using PRIMARY (id=carries.pharmacy\_id) (cost=0.25 rows=1) (actual time=0.00221..0.00226 rows=1 loops=52) -> Select #2 (subquery in condition; run only once) -> Rows fetched before execution (cost=0..0 rows=1) (actual time=75e-6..128e-6 rows=1 loops=1) -> Zero rows (no matching row in const table) (cost=0..0 rows=0) (actual time=83e-6..83e-6 rows=0 loops=1)

Rows per page: 20 ▾ 1 – 1 of 1 |< < > >|

## Which Pharmacies Carry + Pharmacies.city + Users.city:

Run Save Format Clear Syntax error at or near "ANALYZE"

```

1 # check if your local pharmacies + favorite pharmacy carries some medicine
2 EXPLAIN ANALYZE
3 (SELECT medicines.name AS medicine_name, pharmacies.id, pharmacies.state, pharmacies.address, pharmacies.city
4 FROM stage2_schema.medicines AS medicines
5 JOIN stage2_schema.carries AS carries ON medicines.name = carries.medicine_name
6 JOIN stage2_schema.pharmacies AS pharmacies ON pharmacies.id = carries.pharmacy_id
7 WHERE pharmacies.city = (SELECT users.city FROM stage2_schema.users AS users WHERE users.email = 'a.brooks@outlook.com')
8 AND medicines.name = 'A Doxid 100mg Capsule'
9 UNION #automatically removes duplicates
10 (SELECT medicines.name AS medicine_name, pharmacies.id, pharmacies.state, pharmacies.address, pharmacies.city
11 FROM stage2_schema.medicines AS medicines
12 JOIN stage2_schema.carries AS carries ON medicines.name = carries.medicine_name
13 JOIN stage2_schema.pharmacies AS pharmacies ON pharmacies.id = carries.pharmacy_id

```

Results Execution time: 1.3 ms Execution time: 2.7 ms Export ▾

**EXPLAIN**

-> Table scan on <union temporary> (cost=38.4..39.9 rows=2.6) (actual time=0.219..0.22 rows=2 loops=1) -> Union materialize with deduplication (cost=37.4..37.4 rows=2.6) (actual time=0.217..0.217 rows=2 loops=1) -> Nested loop inner join (cost=37.1 rows=2.6) (actual time=0.0246..0.192 rows=2 loops=1) -> Covering index lookup on carries using idx\_med\_pharm (medicine\_name='A Doxid 100mg Capsule') (cost=18.9 rows=52) (actual time=0.0147..0.0524 rows=52 loops=1) -> Filter: (pharmacies.city = (select #2)) (cost=0.25 rows=0.05) (actual time=0.00253..0.00254 rows=0.0385 loops=52) -> Single-row index lookup on pharmacies using PRIMARY (id=carries.pharmacy\_id) (cost=0.25 rows=1) (actual time=0.00222..0.00225 rows=1 loops=52) -> Select #2 (subquery in condition; run only once) -> Rows fetched before execution (cost=0..0 rows=1) (actual time=117e-6..175e-6 rows=1 loops=1) -> Zero rows (no matching row in const table) (cost=0..0 rows=0) (actual time=81e-6..81e-6 rows=0 loops=1)

Rows per page: 20 ▾ 1 – 1 of 1 |< < > >|

## Which Pharmacies Carry + Pharmacies.state + Users.state:

Syntax error at or near "ANALYZE"

```

1 # check if your local pharmacies + favorite pharmacy carries some medicine
2 EXPLAIN ANALYZE
3 (SELECT medicines.name AS medicine_name, pharmacies.id, pharmacies.state, pharmacies.address, pharmacies.city
4 FROM stage2_schema.medicines AS medicines
5 JOIN stage2_schema.carries AS carries ON medicines.name = carries.medicine_name
6 JOIN stage2_schema.pharmacies AS pharmacies ON pharmacies.id = carries.pharmacy_id
7 WHERE pharmacies.city = (SELECT users.city FROM stage2_schema.users AS users WHERE users.email = 'a.brooks@outlook.com')
8 AND medicines.name = 'A Doxid 100mg Capsule'
9 UNION #automatically removes duplicates
10 (SELECT medicines.name AS medicine_name, pharmacies.id, pharmacies.state, pharmacies.address, pharmacies.city
11 FROM stage2_schema.medicines AS medicines
12 JOIN stage2_schema.carries AS carries ON medicines.name = carries.medicine_name
13 JOIN stage2_schema.pharmacies AS pharmacies ON pharmacies.id = carries.pharmacy_id

```

**Results** Execution time: 1.3 ms Execution time: 1.5 ms

**EXPLAIN**

```

-> Table scan on <union temporary> (cost=38.2..40.2 rows=5.2) (actual time=0.196..0.197 rows=2 loops=1) -> Union materialize with deduplication
(cost=37.7..37.7 rows=5.2) (actual time=0.194..0.194 rows=2 loops=1) -> Nested loop inner join (cost=37.1 rows=5.2) (actual time=0.0178..0.183 rows=2
loops=1) -> Covering index lookup on carries using idx_med_pharm (medicine_name='A Doxid 100mg Capsule') (cost=18.9 rows=52) (actual
time=0.00903..0.0433 rows=52 loops=1) -> Filter: (pharmacies.city = (select #2)) (cost=0.25 rows=0.1) (actual time=0.00255..0.00256 rows=0.0385 loops=52) ->
Single-row index lookup on pharmacies using PRIMARY (id=carries.pharmacy_id) (cost=0.25 rows=1) (actual time=0.00222..0.00226 rows=1 loops=52) -> Select
#2 (subquery in condition; run only once) -> Rows fetched before execution (cost=0..0 rows=1) (actual time=72e-6..110e-6 rows=1 loops=1) -> Zero rows (no
matching row in const table) (cost=0..0 rows=0) (actual time=71e-6..71e-6 rows=0 loops=1)

```

Rows per page: 20 ▾ 1 – 1 of 1 |< < > >|

For the which pharmacies carry query, the default cost is 38.2. We experimented with indexes on pharmacy.address, pharmacies.city + users.city, and users.state + pharmacies.state. Among these, indexing on pharmacies.city + users.city slightly increased the cost to 38.4. This minor increase is likely due to the query's use of a UNION operator, which requires a deduplication step involving sorting or hashing across both result sets. Since the additional index adds lookup overhead without improving selectivity, the planner estimates a slightly higher cost. Overall, the default indexing strategy is still the most efficient for this query.