

# ClimaCloset - Weather Based Outfit Suggestion App

Jack Moran, Ben Goldman, Livia Bezati, and Reda Mehdaoui

Stage 3 - Team 085

## UserDatabase

Our database includes 5 tables: users, clothingItems, outfits, weather, and restaurants. These tables are shown below along with the DDL commands for each table.

```
mysql> show tables;
+----------------+
| Tables_in_UserDatabase |
+-----+
| clothingItems          |
| outfits                |
| restaurants             |
| users                  |
| weather                |
+-----+
5 rows in set (0.00 sec)
```

- [clothingItems](#):

```
mysql> select count(*) from clothingItems;
+-----+
| count(*) |
+-----+
|      2000 |
+-----+
1 row in set (0.00 sec)

mysql> describe clothingItems;
+-----+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| clothingId | int        | NO   | PRI | NULL    |       |
| name        | varchar(50) | NO   |     | NULL    |       |
| type        | varchar(50) | NO   |     | NULL    |       |
| style       | varchar(50) | NO   |     | NULL    |       |
| color       | varchar(50) | NO   |     | NULL    |       |
| userId      | int        | YES  | MUL | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.02 sec)
```

DDL Command:

```
CREATE TABLE clothingItems (
    clothingId INT PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    type VARCHAR(50) NOT NULL,
```

```

style VARCHAR(50) NOT NULL,
color VARCHAR(50) NOT NULL,
userId INT,
FOREIGN KEY (userId) REFERENCES users(userId)
);

```

- outfits:

```

mysql> select count(*) from outfits;
+-----+
| count(*) |
+-----+
|      445 |
+-----+
1 row in set (0.02 sec)

mysql> describe outfits;
+-----+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| outfitid   | int        | NO   | PRI | NULL    |       |
| topLayer1  | int        | YES  | MUL | NULL    |       |
| topLayer2  | int        | YES  | MUL | NULL    |       |
| bottom     | int        | YES  | MUL | NULL    |       |
| outfitStyle | varchar(50) | NO   |       | NULL    |       |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.01 sec)

```

DDL Command:

```

CREATE TABLE outfits (
    outfitid INT PRIMARY KEY,
    topLayer1 INT,
    topLayer2 INT,
    bottom INT,
    outfitStyle VARCHAR(50) NOT NULL,
    FOREIGN KEY (topLayer1) REFERENCES clothingItems(clothingId),
    FOREIGN KEY (topLayer2) REFERENCES clothingItems(clothingId),
    FOREIGN KEY (bottom) REFERENCES clothingItems(clothingId)
);

```

- restaurants:

```

mysql> select count(*) from restaurants;
+-----+
| count(*) |
+-----+
|      51717 |
+-----+
1 row in set (0.25 sec)

mysql> describe restaurants;
+-----+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| url        | varchar(255) | YES  |     | NULL    |       |
| address    | varchar(255) | YES  |     | NULL    |       |
| name       | varchar(50)  | YES  |     | NULL    |       |
| online_order | tinyint(1) | YES  |     | NULL    |       |
| book_table  | tinyint(1)  | YES  |     | NULL    |       |
| rate        | decimal(10,2) | YES  |     | NULL    |       |
| votes       | int         | YES  | MUL | NULL    |       |
| phone       | int         | YES  |     | NULL    |       |
| location    | varchar(50) | YES  | MUL | NULL    |       |
| rest_type   | varchar(50) | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
10 rows in set (0.00 sec)

```

DDL Command:

```

CREATE TABLE restaurants (
    url        VARCHAR(255) ,
    address    VARCHAR(255) ,
    name       VARCHAR(50) ,
    online_order TINYINT(1) ,
    book_table  TINYINT(1) ,
    rate        DECIMAL(10,2) ,
    votes       INT ,
    phone       INT ,
    location    VARCHAR(50) ,
    rest_type   VARCHAR(50) ,
);

```

- users:

```

mysql> select count(*) from users;
+-----+
| count(*) |
+-----+
|      100 |
+-----+
1 row in set (0.04 sec)

mysql> describe users;
+-----+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra        |
+-----+-----+-----+-----+-----+-----+
| userId     | int        | NO   | PRI | NULL    | auto_increment |
| username   | varchar(50) | NO   | UNI | NULL    |                |
| password   | varchar(255) | NO   |      | NULL    |                |
| firstName  | varchar(50) | NO   |      | NULL    |                |
| lastName   | varchar(50) | NO   |      | NULL    |                |
| address    | varchar(255) | NO   |      | NULL    |                |
| location   | varchar(100) | NO   |      | NULL    |                |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.02 sec)

```

DDL Command:

```

CREATE TABLE users (
    userId INT PRIMARY KEY AUTO_INCREMENT,
    username VARCHAR(50) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50) NOT NULL,
    address VARCHAR(255) NOT NULL,
    location VARCHAR(100) NOT NULL
);

```

- **weather**:

```

mysql> select count(*) from weather;
+-----+
| count(*) |
+-----+
|    7320 |
+-----+
1 row in set (0.02 sec)

mysql> describe weather;
+-----+-----+-----+-----+-----+-----+
| Field      | Type       | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| station    | varchar(20) | YES  |     | NULL    |       |
| date       | varchar(20) | YES  |     | NULL    |       |
| latitude   | float       | YES  |     | NULL    |       |
| longitude  | float       | YES  |     | NULL    |       |
| elevation  | float       | YES  |     | NULL    |       |
| location   | varchar(100) | YES  |     | NULL    |       |
| month      | int        | YES  |     | NULL    |       |
| day        | int        | YES  |     | NULL    |       |
| hour       | int        | YES  |     | NULL    |       |
| dailyAvg   | float       | YES  |     | NULL    |       |
| dailyAvgMax | float      | YES  |     | NULL    |       |
| dailyAvgMin | float      | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
12 rows in set (0.01 sec)

```

DDL Command:

```

CREATE TABLE weather (
    station VARCHAR(20),
    date VARCHAR(20),
    latitude FLOAT,
    longitude FLOAT,
    elevation FLOAT,
    location VARCHAR(100),
    month INT,
    day INT,
    hour INT,
    dailyAvg FLOAT,
    dailyAvgMax FLOAT,
    dailyAvgMin FLOAT
);

```

## QUERY #1

```

SELECT c.userId, o.outfitStyle, COUNT(*) AS styleCount
FROM outfits o
JOIN clothingItems c ON o.topLayer1 = c.clothingId
GROUP BY c.userId, o.outfitStyle
ORDER BY c.userId, styleCount DESC;

```

This query finds the number of each style of outfits that a user has. This is a query that will potentially be helpful when we try to show a user's closet as we can share how many outfits they can make for each different styles we have.

```

mysql> SELECT c.userId, o.outfitStyle, COUNT(*) AS styleCount FROM outfits o JOIN clothingItems c ON o.topLayer1 = c.clothingId GROUP BY c.userId, o.outfitStyle ORDER BY c.userId, styleCount DESC LIMIT 15;
+-----+-----+-----+
| userId | outfitStyle | styleCount |
+-----+-----+-----+
| 1 | winter | 2 |
| 1 | summer | 2 |
| 1 | athletic | 1 |
| 1 | formal | 2 |
| 2 | casual | 1 |
| 3 | athletic | 3 |
| 3 | formal | 2 |
| 4 | formal | 2 |
| 4 | summer | 1 |
| 4 | athletic | 1 |
| 5 | winter | 3 |
| 5 | business casual | 1 |
| 5 | formal | 1 |
| 6 | winter | 2 |
| 6 | formal | 1 |
| 7 | casual | 1 |
+-----+-----+-----+
15 rows in set (0.22 sec)

mysql> EXPLAIN ANALYZE SELECT c.userId, o.outfitStyle, COUNT(*) AS styleCount
    >> FROM outfits o
    >> JOIN clothingItems c ON o.topLayer1 = c.clothingId
    >> GROUP BY c.userId, o.outfitStyle
    >> ORDER BY c.userId, styleCount DESC
+-----+
| EXPLAIN
+-----+
| >> Sort: c.userId, styleCount DESC (actual time=69.7..69.7 rows=319 loops=1)
|   >> Table scan on <temporary> (actual time=68.5..68.6 rows=319 loops=1)
|     >> Aggregate using temporary table (actual time=68.5..68.5 rows=319 loops=1)
|       >> Nested loop inner join (cost=499 rows=445) (actual time=69.7..67.6 rows=445 loops=1)
|         >> Filter: (o.topLayer1 IS NOT NULL) (cost=45.2 rows=445) (actual time=41.2..43.9 rows=445 loops=1)
|           >> Table scan on o (cost=46.4 rows=445) (actual time=41.1..43.9 rows=445 loops=1)
|             >> Single-row index lookup on c using PRIMARY (clothingId=o.topLayer1) (cost=0.917 rows=1) (actual time=0.0518..0.0519 rows=1 loops=445)
+-----+
1 row in set (0.11 sec)

```

Index #1:

```
CREATE INDEX idx_toplayer1 ON outfits(topLayer1);
```

```

mysql> EXPLAIN ANALYZE SELECT c.userId, o.outfitStyle, COUNT(*) AS styleCount FROM outfits o JOIN clothingItems c ON o.topLayer1 = c.clothingId GROUP BY c.userId, o.outfitStyle ORDER BY c.userId, styleCount DESC
+-----+
| EXPLAIN
+-----+
| >> Sort: c.userId, styleCount DESC (actual time=1.4..1.4 rows=319 loops=1)
|   >> Table scan on <temporary> (actual time=1.24..1.24 rows=319 loops=1)
|     >> Aggregate using temporary table (actual time=1.24..1.24 rows=319 loops=1)
|       >> Nested loop inner join (cost=201 rows=445) (actual time=1.143..0.876 rows=445 loops=1)
|         >> Filter: (o.topLayer1 IS NOT NULL) (cost=45.2 rows=445) (actual time=0.123..0.277 rows=445 loops=1)
|           >> Table scan on o (cost=46.2 rows=445) (actual time=0.122..0.246 rows=445 loops=1)
|             >> Single-row index lookup on c using PRIMARY (clothingId=o.topLayer1) (cost=0.23 rows=1) (actual time=0.00108..0.00111 rows=1 loops=445)
+-----+
1 row in set (0.01 sec)

```

Index #2:

```
DROP INDEX idx_toplayer1 ON outfits;
CREATE INDEX idx_userid ON clothingItems(userId);
```

```

mysql> EXPLAIN ANALYZE SELECT c.userId, o.outfitStyle, COUNT(*) AS styleCount FROM outfits o JOIN clothingItems c ON o.topLayer1 = c.clothingId GROUP BY c.userId, o.outfitStyle ORDER BY c.userId, styleCount DESC;
+-----+
| EXPLAIN
+-----+
| > Sort: c.userId, styleCount DESC (actual time=1.15..1.21 rows=319 loops=1)
|   > Table scan on <temporary> (actual time=1.16..1.21 rows=319 loops=1)
|     > Aggregate using temporary table (actual time=1.16..1.16 rows=319 loops=1)
|       > Nested loop inner join (cost=201 rows=45) (actual time=0.0685..0.072 rows=445 loops=1)
|         > Filter: (o.topLayer1 is not null) (cost=45.2 rows=45) (actual time=0.051..0.051 rows=445 loops=1)
|           > Table scan on o (cost=45.2 rows=45) (actual time=0.0006..0.0171 rows=445 loops=1)
|             > Single-row index lookup on o using PRIMARY (clothingId=o.topLayer1) (cost=0.25 rows=1) (actual time=0.00107..0.0011 rows=1 loops=445)
|
+-----+
1 row in set (0.01 sec)

```

Index #3:

```

DROP INDEX idx_userid ON clothingItems;
CREATE INDEX idx_id_user ON clothingItems(clothingId, userId);

```

```

mysql> EXPLAIN ANALYZE SELECT c.userId, o.outfitStyle, COUNT(*) AS styleCount FROM outfits o JOIN clothingItems c ON o.topLayer1 = c.clothingId GROUP BY c.userId, o.outfitStyle ORDER BY c.userId, styleCount DESC;
+-----+
| EXPLAIN
+-----+
| > Sort: c.userId, styleCount DESC (actual time=1.45..1.47 rows=319 loops=1)
|   > Table scan on <temporary> (actual time=1.32..1.37 rows=319 loops=1)
|     > Aggregate using temporary table (actual time=1.32..1.37 rows=319 loops=1)
|       > Nested loop inner join (cost=201 rows=45) (actual time=0.921..0.921 rows=445 loops=1)
|         > Filter: (o.topLayer1 is not null) (cost=45.2 rows=45) (actual time=0.109..0.273 rows=445 loops=1)
|           > Table scan on o (cost=45.2 rows=45) (actual time=0.106..0.235 rows=445 loops=1)
|             > Single-row index lookup on o using PRIMARY (clothingId=o.topLayer1) (cost=0.25 rows=1) (actual time=0.00124..0.00127 rows=1 loops=445)
|
+-----+
1 row in set (0.03 sec)

```

	Cost of intended change before
Default	499
idx_toplayer1	201
idx_toplayer1	201
idx_userid	201

We put indexes on the outfits and clothing items tables(join clause and group by). We found a decrease in all of the indexes by the same amount. We ended up deciding that our idx\_userid index was the best fitting for this.

## QUERY #2

```

SELECT c.userId, o.outfitStyle, COUNT(*) AS styleCount
FROM outfits o
JOIN clothingItems c ON o.topLayer1 = c.clothingId
GROUP BY c.userId, o.outfitStyle
HAVING COUNT(*) = (
    SELECT MAX(style_count)

```

```

FROM (
    SELECT COUNT(*) AS style_count
    FROM outfits o2
    JOIN clothingItems c2 ON o2.topLayer1 = c2.clothingId
    WHERE c2.userId = c.userId
    GROUP BY o2.outfitStyle
) sub
);

```

This query returns the most common outfit styles that a user has. This will be helpful to determine what style may be the user's favorite within their closet, and can help when recommending outfits.

```

mysql> SELECT c.userId, o.outfitStyle, COUNT(*) AS styleCount
    FROM outfits o
    JOIN clothingItems c ON o.topLayer1 = c.clothingId
    GROUP BY c.userId, o.outfitStyle HAVING COUNT(*) = (
        SELECT MAX(style_count)
        FROM (
            SELECT COUNT(*) AS style_count
            FROM outfits o2
            JOIN clothingItems c2 ON o2.topLayer1 = c2.clothingId
            WHERE c2.userId = c.userId
            GROUP BY o2.outfitStyle
        ) sub
    );
15 rows in set (0.01 sec)

mysql> EXPLAIN ANALYZE SELECT c.userId, o.outfitStyle, COUNT(*) AS styleCount
    FROM outfits o
    JOIN clothingItems c ON o.topLayer1 = c.clothingId
    GROUP BY c.userId, o.outfitStyle HAVING COUNT(*) = (
        SELECT MAX(style_count)
        FROM (
            SELECT COUNT(*) AS style_count
            FROM outfits o2
            JOIN clothingItems c2 ON o2.topLayer1 = c2.clothingId
            WHERE c2.userId = c.userId
            GROUP BY o2.outfitStyle
        ) sub
    );
+----+-----+
| EXPLAIN
+----+-----+
4--+
+----+-----+
|   > Filter: ('count(G)' = '(select #2)')
|   > Table scan on <temporary> (actual time=22.8..22.8 rows=319 loops=1)
|       > Aggregate using temporary table (actual time=22.8..22.8 rows=319 loops=1)
|           > Nested loop inner join (cost=201 rows=45) (actual time=0.193..0.195 rows=445 loops=1)
|               > Table scan on o (cost=0.01 rows=319 loops=1)
|                   > Table scan on c (cost=40.2 rows=445) (actual time=0.133..0.362 rows=445 loops=1)
|                       > Single-row index lookup on c using PRIMARY (clothingId=o.topLayer1) (cost=0.25 rows=1) (actual time=0.0022..0.00223 rows=1 loops=445)
+> Select #2 (subquery in projection; dependent)
|   > Aggregate using temporary count (actual time=2.5..2.5 rows=1) (actual time=0.061..0.061 rows=1 loops=319)
|       > Table scan on o (cost=2.5..2.5 rows=0) (actual time=0.0598..0.0602 rows=3.38 loops=319)
|           > Materialize (cost=0..0 rows=0) (actual time=0.0596..0.0598 rows=3.38 loops=319)
|               > Table scan on <temporary> (actual time=0.0578..0.0582 rows=3..38 loops=319)
|                   > Aggregated nested inner join (cost=11.5 rows=25..3) (actual time=0.0243..0.0517 rows=20 loops=319)
|                       > Covering index lookup on c2 using idx_userid (userId=c.userId) (cost=2.64 rows=20) (actual time=0.0108..0.0133 rows=20 loops=319)
|                           > Index lockup on o2 using idx_toplayer1 (topLayer1=c2.clothingId) (cost=0.322 rows=1..26) (actual time=0.00168..0.00178 rows=0..229 loops=6380)
+>
+----+-----+
1 row in set, 2 warnings (0.06 sec)

```

Index #1:

```

CREATE INDEX idx_toplayer1 ON outfits(topLayer1);
DROP INDEX idx_toplayer1 ON outfits;

```

```

| -> Filter: ('count(0)' = '(select #2)') (actual time=17.3..17.4 rows=159 loops=1)
-> Table scan on <temporary> (actual time=17.3..17.4 rows=319 loops=1)
-> Aggregate using temporary table (actual time=17.3..17.4 rows=319 loops=1)
-> Nested loop inner join (cost=201 rows=445) (actual time=17.3..17.4 rows=319 loops=1)
-> Filter: (o.toplayer1 is not null) (cost=45.2 rows=445) (actual time=0.0919..0.302 rows=445 loops=1)
-> Single-row index lookup on c using PRIMARY (clothingId=o.toplayer1) (cost=0.25 rows=1 loops=445)
-> Select #2 (subquery in projection; dependent)
-> Aggregate: max(styleCount) (cost=2.5..2.5 rows=1) (actual time=0.0483..0.0483 rows=1 loops=319)
-> Table scan on sub (cost=2.5..2.5 rows=0) (actual time=0.0472..0.0475 rows=3..38 loops=319)
-> Materialize (cost=0..0 rows=0) (actual time=0.0465..0.0469 rows=3..38 loops=319)
-> Table scan on <temporary> (actual time=0.0465..0.0469 rows=3..38 loops=319)
-> Aggregate using temporary table (actual time=0.0454..0.0454 rows=3..38 loops=319)
-> Nested loop inner join (cost=11.5 rows=25..3) (actual time=0.014..0.041 rows=4..57 loops=319)
-> Covering index lookup on c2 using idx_userid (userId=c.userid) (cost=2.64 rows=20) (actual time=0.00505..0.00764 rows=20 loops=319)
-> Index lookup on o2 using idx_toplayer1 (toplayer1=c2.clothingId) (cost=0.322 rows=1..26) (actual time=0.00143..0.00153 rows=0..229 loops=6380)
|

```

## Index #2:

CREATE INDEX idx\_id ON clothingItems(clothingId);  
DROP INDEX idx\_id ON clothingItems;

```

| -> Filter: ('count(0)' = '(select #2)') (actual time=277..277 rows=159 loops=1)
-> Table scan on <temporary> (actual time=277..277 rows=319 loops=1)
-> Aggregate using temporary table (actual time=277..277 rows=319 loops=1)
-> Nested loop inner join (cost=462 rows=445) (actual time=152..183 rows=445 loops=1)
-> Filter: (o.toplayer1 is not null) (cost=462 rows=445) (actual time=152..183 rows=445 loops=1)
-> Table scan on sub (cost=2.5..2.5 rows=0) (actual time=81..82.8 rows=445 loops=445)
-> Single-row index lookup on c using PRIMARY (clothingId=o.toplayer1) (cost=0..0.834 rows=1 loops=445)
-> Select #2 (subquery in projection; dependent)
-> Aggregate: max(styleCount) (cost=2.5..2.5 rows=1) (actual time=0.283..0.283 rows=1 loops=319)
-> Table scan on sub (cost=2.5..2.5 rows=0) (actual time=0.282..0.282 rows=3..38 loops=319)
-> Materialize (cost=0..0 rows=0) (actual time=0.282..0.282 rows=3..38 loops=319)
-> Table scan on <temporary> (actual time=0.28..0.28 rows=3..38 loops=319)
-> Aggregate using temporary table (actual time=0.28..0.28 rows=3..38 loops=319)
-> Nested loop inner join (cost=11.5 rows=25..3) (actual time=0.041..0.041 rows=4..57 loops=319)
-> Covering index lookup on c2 using idx_userid (userId=c.userid) (cost=3.02 rows=20) (actual time=0.201..0.204 rows=20 loops=319)
-> Index lookup on o2 using idx_toplayer1 (toplayer1=c2.clothingId) (cost=0.96 rows=1..26) (actual time=0.00325..0.00335 rows=0..229 loops=6380)
|

```

## Index #3:

CREATE INDEX idx\_userid\_clothingid ON clothingItems(userId, clothingId);  
DROP INDEX idx\_userid\_clothingid ON clothingItems;

```

| -> Filter: ('count(0)' = '(select #2)') (actual time=16.9..17 rows=159 loops=1)
-> Table scan on <temporary> (actual time=16.9..17 rows=319 loops=1)
-> Aggregate using temporary table (actual time=16.9..16.9 rows=319 loops=1)
-> Nested loop inner join (cost=201 rows=445) (actual time=0.188..1.26 rows=445 loops=1)
-> Filter: (o.toplayer1 is not null) (cost=45.2 rows=445) (actual time=0.446..0.443 rows=445 loops=1)
-> Table scan on sub (cost=2.5..2.5 rows=0) (actual time=0.143..0.382 rows=445 loops=1)
-> Single-row index lookup on c using PRIMARY (clothingId=o.toplayer1) (cost=0.25 rows=1) (actual time=0.00166..0.00169 rows=1 loops=445)
-> Select #2 (subquery in projection; dependent)
-> Aggregate: max(styleCount) (cost=2.5..2.5 rows=1) (actual time=0.446..0.446 rows=1 loops=319)
-> Table scan on sub (cost=2.5..2.5 rows=0) (actual time=0.445..0.445 rows=3..38 loops=319)
-> Materialize (cost=0..0 rows=0) (actual time=0.445..0.445 rows=3..38 loops=319)
-> Table scan on <temporary> (actual time=0.441..0.4445 rows=3..38 loops=319)
-> Aggregate using temporary table (actual time=0.441..0.4445 rows=3..38 loops=319)
-> Nested loop inner join (cost=11.5 rows=25..3) (actual time=0.014..0.0396 rows=4..57 loops=319)
-> Covering index lookup on c2 using idx_userid (userId=c.userid) (cost=2.64 rows=20) (actual time=0.00562..0.0081 rows=20 loops=319)
-> Index lookup on o2 using idx_toplayer1 (toplayer1=c2.clothingId) (cost=0..322 rows=1..26) (actual time=0.00136..0.00145 rows=0..229 loops=6380)
|

```

1 row in set, 2 warnings (0.04 sec)

	Cost
Default	201

idx_toplayer1	201
idx_id	462
idx_userid_clothingid	201

We tried indexes on the clothingItems and outfits tables, as well as both of them together(on their join and group by clauses). This did not benefit the cost at all and actually was harmful in some cases. We ultimately decided that the default index worked the best.

### **QUERY #3**

```

SELECT u.userId, o.outfitId, t1.name AS topLayer1_name, t2.name AS topLayer2_name,
b.name AS bottom_name, w.dailyAvg, w.location
FROM users u
JOIN weather w ON u.location = w.location AND w.month = 1 AND w.day = 1
JOIN outfits o ON o.outfitId =
    SELECT o2.outfitId
    FROM outfits o2
    JOIN clothingItems t1 ON o2.topLayer1 = t1.clothingId AND t1.userId = u.userId
    LEFT JOIN clothingItems t2 ON o2.topLayer2 = t2.clothingId AND t2.userId = u.userId
    JOIN clothingItems b ON o2.bottom = b.clothingId AND b.userId = u.userId
    WHERE (
        (w.dailyAvg < 35 AND (t1.type = 'jacket' OR IFNULL(t2.type, '') = 'jacket')) OR
        (w.dailyAvg >= 35 AND w.dailyAvg < 45 AND (t1.type = 'long sleeve' OR
IFNULL(t2.type, '') = 'long sleeve')) OR
        (w.dailyAvg >= 45 AND (
            t1.type IN ('jacket', 'long sleeve') OR
            IFNULL(t2.type, '') IN ('jacket', 'long sleeve')
        ))
    )
LIMIT 1
)
JOIN clothingItems t1 ON o.topLayer1 = t1.clothingId AND t1.userId = u.userId
LEFT JOIN clothingItems t2 ON o.topLayer2 = t2.clothingId AND t2.userId = u.userId
JOIN clothingItems b ON o.bottom = b.clothingId AND b.userId = u.userId
ORDER BY userId ASC;

```

This is a more advanced query that selects a user's outfit based on the weather, at their location, at that specific date. If the weather is less than 35 degrees, it picks an outfit with a jacket. If the weather is within the range of 35 to 45 degrees, it picks an outfit with a jacket or long sleeve. If it

is greater than 45, it will select any outfit. This will be a very helpful query to actually determine what outfit to recommend to the user based on the weather.

```
mysql> SELECT u.userid, o.outfitId, tl.name AS topLayer1_name, t2.name AS topLayer2_name, b.name AS bottom_name, w.location FROM users u JOIN weather w ON u.location = w.location AND month = 1 AND w.day = 1 JOIN outfits o ON o.outfitId = (SELECT o2.outfitId WHERE o2.outfitName = 'outfit1') JOIN clothingItems t1 ON o2.topLayer1 = t1.clothingId AND t1.userId = u.userId WHERE ((w.dailyAvg >= 35 AND t1.type = 'jacket' OR IFNULL(t2.type, '') = 'jacket') OR (w.dailyAvg < 35 AND t1.type = 'long sleeve' OR IFNULL(t2.type, '') = 'long sleeve')) OR (w.dailyAvg >= 45 AND (t1.type IN ('jacket', 'long sleeve') OR IFNULL(t2.type, '') IN ('jacket', 'long sleeve'))) LIMIT 1) JOIN clothingItems t1 ON o.topLayer1 = t1.clothingId AND t1.userId = u.userId LEFT JOIN clothingItems t2 ON o.topLayer2 = t2.clothingId AND t2.userId = u.userId ORDER BY userId ASC LIMIT 1;
+-----+-----+-----+-----+-----+-----+
| userid | outfitId | topLayer1_name | topLayer2_name | bottom_name | location |
+-----+-----+-----+-----+-----+-----+
| 1 | 7001 | green dress shirt | green jacket | yellow jeans | 34.5 | New York, NY |
| 2 | 7006 | red sweater | gray jacket | beige jeans | 46.2 | Dallas, TX |
| 3 | 7010 | green long sleeve | beige rain coat | white pants | 50.6 | San Francisco, CA |
| 4 | 7015 | gray long sleeve | NULL | red jeans | 50.6 | San Francisco, CA |
| 5 | 7019 | blue long sleeve | blue rain coat | grey jeans | 49.5 | Las Vegas, NV |
| 6 | 7024 | brown long sleeve | beige rain coat | black jeans | 46.5 | Seattle, WA |
| 7 | 7029 | brown long sleeve | red jacket | red jeans | 41.9 | Seattle, WA |
| 8 | 7038 | blue long sleeve | NULL | blue pants | 60.2 | Los Angeles, CA |
| 9 | 7040 | white long sleeve | white rain coat | white pants | 50.2 | Phoenix, AZ |
| 10 | 7044 | brown long sleeve | red rain coat | white pants | 60.2 | Los Angeles, CA |
| 11 | 7053 | brown long sleeve | green jacket | blue pants | 46.2 | Dallas, TX |
| 12 | 7064 | green dress shirt | gray jacket | brown pants | 30.7 | Detroit, MI |
| 13 | 7070 | green dress shirt | brown jacket | yellow jeans | 70.1 | Boston, MA |
| 14 | 7074 | brown dress shirt | beige jacket | black jeans | 46.2 | Dallas, TX |
| 15 | 7084 | brown dress shirt | white jacket | white pants | 17.9 | Minneapolis, MN |
| 20 | 7086 | yellow dress shirt | white jacket | blue pants | 17.9 | Minneapolis, MN |
+-----+-----+-----+-----+-----+-----+
15 rows in set (0.02 sec)
```

```
| -> Sort: userid (actual_time=333.333 rows=72 loops=1)
-> Stream results (cost=1226 rows=0.164) (actual_time=268..332 rows=72 loops=1)
  -> Filter: (o.outfitId = selected(2)) AND o.location = w.location AND (cost=1226 rows=0.164) (actual_time=268..332 rows=72 loops=1)
    -> InnoDB hash join (char(o.location),<derived>,w.location)
      -> Filter: ((w.`day` = 1) AND (w.`month` = 1)) (cost=36.9 rows=7361) (actual_time=127..175 rows=20 loops=1)
        -> Table scan on w (cost=36.9 rows=7361) (actual_time=127..175 rows=7320 loops=1)
      -> Hash
        -> Nested loop left join (cost=389 rows=22..3) (actual_time=11..11.11 rows=445 loops=1)
          -> Nested loop inner join (cost=381 rows=22..3) (actual_time=11..11.11 rows=445 loops=1)
            -> Nested loop inner join (cost=357 rows=22..3) (actual_time=0.856..2.63 rows=445 loops=1)
              -> Nested loop inner join (cost=201 rows=445) (actual_time=0.844..1.39 rows=445 loops=1)
                -> Filter: (t1.userId = t1.userId) (cost=0.25 rows=445) (actual_time=0.816..1.02 rows=445 loops=1)
                  -> Table scan on t1 (cost=45.2 rows=445) (actual_time=0.111..0.256 rows=445 loops=1)
                    -> Filter: (t1.userId is not null) (cost=0.25 rows=1) (actual_time=0.00191..0.002 rows=445)
                      -> Single-row index lookup on t1 using PRIMARY (clothingId=o.topLayer1) (cost=0.25 rows=1) (actual_time=0.00176..0.00179 rows=1 loops=445)
                    -> Filter: (t1.userId = t1.userId) (cost=0.25 rows=1) (actual_time=0.00102..0.00105 rows=1 loops=445)
                      -> Single-row index lookup on b using PRIMARY (clothingId=o.bottom) (cost=1 rows=1) (actual_time=0.00128..0.00131 rows=1 loops=445)
                    -> Filter: (t2.userId = t1.userId) (cost=0.254 rows=1) (actual_time=0.0104..0.0011 rows=0.706 loops=445)
                      -> Single-row index lookup on t2 using PRIMARY (clothingId=o.topLayer2) (cost=0.254 rows=1) (actual_time=849e-6..869e-6 rows=0.706 loops=445)
        -> Select #2 (subquery condition: dependent)
          -> Limit: 1 row(s) (cost=22.3 rows=1) (actual_time=0.0872..0.0872 rows=0.772 loops=426)
            -> Filter: (((w.dailyAvg < 35) AND (t1.type = 'jacket')) OR ((w.dailyAvg > 35) AND (t1.type = 'long sleeve') OR (IFNULL(t2.type, '') = 'long sleeve')) OR ((w.dailyAvg < 45) AND (t1.type = 'jacket') OR ((w.dailyAvg < 45) AND (t1.type = 'long sleeve')))) (cost=22.3 rows=2..68) (actual_time=0.087..0.087 rows=0.772 loops=426)
              -> Nested loop left join (cost=21..2.58) (actual_time=0.715..0.854 rows=2..3 loops=246)
                -> Nested loop inner join (cost=21..4 rows=2..68) (actual_time=0.0703..0.0825 rows=2..3 loops=246)
                  -> Nested loop inner join (cost=21..2 rows=2..68) (actual_time=0.0679..0.0782 rows=2..3 loops=426)
                    -> Filter: (w.dailyAvg < 45) AND (t1.type = 'jacket') (cost=2..64 rows=20) (actual_time=0.00693..0.00952 rows=9..59 loops=426)
                      -> Covering index lookup on b using bottom (bottom=b.clothingId) (cost=0..342 rows=1..34) (actual_time=0.00695..0.00702 rows=24..1 loops=4094)
                    -> Filter: (t1.userId = t1.userId) (cost=0..342 rows=1..34) (actual_time=0.00695..0.00702 rows=24..1 loops=4094)
                      -> Filter: ((w.dailyAvg < 35) OR ((w.dailyAvg > 35) AND (w.userId = t1.userId)) (cost=0..25 rows=0..1) (actual_time=0.00161..0.00168 rows=1 loops=979)
                        -> Single-row index lookup on t1 using PRIMARY (clothingId=o.topLayer1) (cost=0..251 rows=1) (actual_time=0.00125..0.00127 rows=1 loops=979)
                        -> Filter: (t2.userId = t1.userId) (cost=0..287 rows=1) (actual_time=974e-6..0.00103 rows=0..728 loops=979)
                          -> Single-row index lookup on t2 using PRIMARY (clothingId=o.topLayer2) (cost=0..287 rows=1) (actual_time=805e-6..817e-6 rows=0..728 loops=979)
| s=979)
```

## Index #1:

```
CREATE INDEX idx_loc_date ON weather(location, month, day);
DROP INDEX idx_date ON weather;
```

```
| -> Nested loop left join (cost=162 rows=0.251) (actual_time=3..53..3.53..3.53..rows=72 loops=1)
  -> Nested loop inner join (cost=162 rows=0.251) (actual_time=32..3..53..3.53..3.53..rows=72 loops=1)
    -> Nested loop inner join (cost=158 rows=5..03) (actual_time=32..1..53..3.53..3.53..rows=72 loops=1)
      -> Nested loop inner join (cost=80..6 rows=10..1) (actual_time=32..1..53..1 rows=72 loops=1)
        -> Nested loop inner join (cost=10..2 rows=10..1) (actual_time=0.26..0.333 rows=100 loops=1)
          -> Index lookup on w using PRIMARY (cost=10..2 rows=100) (actual_time=0.26..0.333 rows=100 loops=1)
            -> Filter: (o.outfitId = selected(#2)) AND (o.topLayer1 is not null) AND (o.bottom is not null) (cost=0..25 rows=1) (actual_time=0.538..0.539 rows=0..75 loops=96)
              -> Single-row index lookup on t1 using PRIMARY (clothingId=o.outfitId) (cost=0..251 rows=1) (actual_time=0..511..0..517 rows=0..75 loops=96)
                -> Select #2 (subquery condition: dependent)
                  -> Limit: 1 row(s) (cost=34..6 rows=1) (actual_time=0..214..0..214 rows=0..9 loops=240)
                    -> Filter: (((w.dailyAvg < 35) AND (t1.type = 'jacket')) OR ((w.dailyAvg < 45) AND (t1.type = 'long sleeve')) OR (IFNULL(t2.type, '') = 'long sleeve')) OR ((w.dailyAvg < 45) AND (t1.type = 'jacket') OR ((w.dailyAvg < 45) AND (t1.type = 'long sleeve')))) (cost=34..6 rows=2..68) (actual_time=0..214..0..214 rows=0..9 loops=240)
                      -> Nested loop left join (cost=34..6 rows=2..68) (actual_time=0..202..0..214 rows=1..9 loops=240)
                        -> Nested loop inner join (cost=32..6 rows=2..68) (actual_time=0..20..0..225 rows=1..9 loops=240)
                          -> Filter: (t1.userId = t1.userId) (cost=0..155..0..0225 rows=1..9 loops=240)
                            -> Covering index lookup on t1 using PRIMARY (clothingId=o.topLayer1) (cost=0..155..0..0225 rows=1..9 loops=240)
                              -> Filter: (t2.userId = t1.userId) (cost=0..342 rows=1..34) (actual_time=0..00177..0..00178 rows=0..238 loops=1920)
                                -> Index lookup on o2 using bottom (bottom=b.clothingId) (cost=0..342 rows=1..34) (actual_time=0..00357..0..00163 rows=0..238 loops=1920)
                                  -> Filter: ((w.dailyAvg < 35) OR ((w.dailyAvg < 45) AND (w.userId = t1.userId)) (cost=0..667 rows=0..1) (actual_time=0..0975..0..08975 rows=1 loops=457)
                                    -> Single-row index lookup on t1 using PRIMARY (clothingId=o2.topLayer1) (cost=0..667 rows=1) (actual_time=0..097..0..0971 rows=1 loops=457)
                                    -> Filter: (t2.userId = t1.userId) (cost=0..704 rows=1) (actual_time=0..00172..0..00175 rows=0..737 loops=457)
                                      -> Single-row index lookup on t1 using PRIMARY (clothingId=o2.topLayer2) (cost=0..704 rows=1) (actual_time=0..00151..0..00152 rows=0..737 loops=457)
                                        -> Filter: (t1.userId = t1.userId) (cost=0..667 rows=0..05) (actual_time=0..00235..0..00236 rows=0..05 loops=72)
                                          -> Single-row index lookup on b using PRIMARY (clothingId=o.bottom) (cost=0..668 rows=1) (actual_time=0..00392..0..00394 rows=1 loops=72)
                                            -> Filter: (t2.userId = t1.userId) (cost=0..66..0..05 rows=1) (actual_time=0..00104..0..00107 rows=0..917 loops=72)
| l=1 loops=457)
```

## Index #2:

```
CREATE INDEX idx_outfitid ON outfits(outfitId);
DROP INDEX idx_outfitid ON outfits;
```

```

| -> Sort: u.userId (actual time=56.56 rows=72 loops=1)
-> Stream results (cost=1361 rows=0.164) (actual time=36.1..55.9 rows=72 loops=1)
  -> Filter: ((t1.userId = t2.userId) AND (t1.location <= u.location)) (cost=1361 rows=0.164) (actual time=36.1..55.9 rows=72 loops=1)
    -> Inner hash join (<hash>(w.location)<hash>(u.location)) (cost=1361 rows=0.164) (actual time=35.9..41.4 rows=426 loops=1)
      -> Table scan on w (cost=36.9 rows=7361) (actual time=0.0218..4.49 rows=732 loops=1)
    -> Hash
      -> Nested loop left join (cost=524 rows=22..3) (actual time=0.11..35.6 rows=445 loops=1)
        -> Nested loop inner join (cost=513 rows=22..3) (actual time=0.107..35.1 rows=445 loops=1)
          -> Nested loop inner join (<hash>(o.userid)<hash>(o.location)) (cost=513 rows=22..3) (actual time=0.0942..34.7 rows=445 loops=1)
            -> Filter: ((o.toplayer1 is not null) and (o.bottom1 is not null)) (cost=45.2 rows=445) (actual time=0.0682..0.341 rows=445 loops=1)
              -> Table scan on o (cost=45.2 rows=445) (actual time=0.0654..0.26 rows=445 loops=1)
              -> Single-row index lookup on b using PRIMARY (clothingId=o.bottom) (cost=0.417 rows=1) (actual time=0.0104..0.0104 rows=1 loops=445)
                -> Filter: (b.userId = t1.userId) (cost=0.417 rows=0.05) (actual time=0.0066..0.0066 rows=1 loops=445)
                -> Single-row index lookup on u using PRIMARY (userId=t1.userId) (cost=0.254 rows=1) (actual time=0.0658..0.0658 rows=1 loops=445)
                  -> Filter: (u.location <= t1.location) (cost=0.421 rows=1) (actual time=91e-6..982e-6 rows=0.706 loops=445)
                    -> Single-row index lookup on t2 using PRIMARY (clothingId=o.toplayer2) (cost=0.421 rows=1) (actual time=767e-6..787e-6 rows=0.706 loops=445)

-> Select #2 (subquery in condition: dependent)
  -> Limit: 1 row(s) (cost=27.3 rows=1) (actual time=0.033..0.033 rows=0.772 loops=426)
    -> Filter: ((w.dailyAvg < 45) and ((t1.type = 'jacket') or ((t1.type = 'long sleeve') or (ifnull(t2.type,'') = 'long sleeve')))) or ((w.dailyAvg > 45) and ((t1.type = 'jacket') or ((t1.type = 'long sleeve') or (ifnull(t2.type,'') = 'long sleeve')))) or ((w.dailyAvg < 35) or ((t1.type = 'jacket') or ((t1.type = 'long sleeve') or (ifnull(t2.type,'') = 'long sleeve'))))
      -> Nested loop left join (cost=27.3 rows=2.68) (actual time=0.018..0.0316 rows=2..3 loops=426)
        -> Nested loop inner join (cost=25.9 rows=2.68) (actual time=0.0167..0.0289 rows=2..3 loops=426)
          -> Nested loop inner join (<hash>(t1.userId)<hash>(t1.location)) (cost=25.9 rows=2.68) (actual time=0.0167..0.0289 rows=2..3 loops=426)
            -> Filter: ((t1.userId = t2.userId) and (t1.location <= t2.location)) (cost=2.64 rows=20) (actual time=0.00506..0.00759 rows=9.59 loops=426)
              -> Covering index lookup on b using idx userid_userid (cost=2.64 rows=20) (actual time=0.00472..0.00581 rows=9.59 loops=426)
                -> Filter: (o2.toplayer1 is not null) (cost=0.342 rows=1..34) (actual time=0.00162..0.0017 rows=2..24 loops=4084)
                -> Index lookup on o using bottom (bottom=o.clothingId) (cost=0.342 rows=1..34) (actual time=0.00149..0.00155 rows=0.24 loops=4084)
                  -> Filter: ((w.dailyAvg < 35) or ((w.dailyAvg > 45) or (w.dailyAvg < 45) and (t1.userId = t2.userId))) (cost=0.417 rows=0.1) (actual time=0.0014..0.00146 rows=1 loops=1)
                    -> Single-row index lookup on t1 using PRIMARY (clothingId=o2.topLayer1) (cost=0.417 rows=1) (actual time=0.00106..0.00108 rows=1 loops=979)
                    -> Filter: (t2.userId = t1.userId) (cost=0.454 rows=1) (actual time=913e-6..959e-6 rows=0.728 loops=979)
                      -> Single-row index lookup on t2 using PRIMARY (clothingId=o2.toplayer2) (cost=0.454 rows=1) (actual time=753e-6..765e-6 rows=0.728 loops=979)

s=979)

```

### Index #3:

CREATE INDEX idx\_userid\_type ON clothingItems(userId, type);  
DROP INDEX idx\_userid\_type ON clothingItems;

```

-----+
| -> Sort: u.userId (actual time=23.3..23.3 rows=72 loops=1)
-> Stream results (cost=1361 rows=0.164) (actual time=23.3..23.3 rows=72 loops=1)
  -> Filter: ((o.outfitid = (select #2)) and (w.location = u.location)) (cost=1357 rows=0.164) (actual time=23.3..23.3 rows=72 loops=1)
    -> Inner hash join (<hash>(w.location)<hash>(u.location)) (cost=1357 rows=0.164) (actual time=23.3..23.3 rows=72 loops=1)
      -> Filter: ((w.day = 1) and (w.month = 1)) (cost=36.9 rows=3761) (actual time=0.0187..5.55 rows=20 loops=1)
        -> Table scan on w (cost=36.9 rows=7361) (actual time=0.0161..4.97 rows=7320 loops=1)
    -> Hash
      -> Nested loop left join (cost=520 rows=22..3) (actual time=0.11..118..2.22 rows=445 loops=1)
        -> Nested loop inner join (cost=512 rows=22..3) (actual time=0.113..117..76 rows=445 loops=1)
          -> Nested loop inner join (cost=494 rows=22..3) (actual time=0.104..104..0.00001 rows=445 loops=1)
            -> Nested loop inner join (cost=201 rows=445) (actual time=0.05..0.87 rows=445 loops=1)
              -> Filter: ((o.toplayer1 is not null) and (o.bottom1 is not null)) (cost=45.2 rows=445) (actual time=0.0542..0.239 rows=445 loops=1)
                -> Table scan on o (cost=45.2 rows=445) (actual time=0.0504..0.188 rows=445 loops=1)
                -> Filter: ((t1.userId = t2.userId) and (t1.location <= t2.location)) (cost=2.64 rows=20) (actual time=0.00101..0.00104 rows=1 loops=445)
                  -> Single-row index lookup on t1 using PRIMARY (clothingId=o.toplayer1) (cost=0.25 rows=1) (actual time=0.00101..0.00104 rows=1 loops=445)
                  -> Single-row index lookup on u using PRIMARY (userId=t1.userId) (cost=0.25 rows=1) (actual time=420e-6..490e-6 rows=1 loops=445)
                    -> Filter: (t2.userId = t1.userId) (cost=0.25 rows=0..05) (actual time=0.0105..0.00114 rows=1 loops=445)
                    -> Single-row index lookup on b using bottom (bottom=o.clothingId) (cost=0.25 rows=1) (actual time=900e-6..928e-6 rows=1 loops=445)
                      -> Filter: ((w.dailyAvg < 35) or ((w.dailyAvg > 45) and (t1.userId = t2.userId))) (cost=0.25 rows=1) (actual time=648e-6..668e-6 rows=0.706 loops=445)
                    -> Single-row index lookup on t2 using PRIMARY (clothingId=o.toplayer2) (cost=0.25 rows=1) (actual time=648e-6..668e-6 rows=0.706 loops=445)

-> Select #2 (subquery in condition: dependent)
  -> Limit: 1 row(s) (cost=22.3 rows=1) (actual time=0.033..0.033 rows=0.772 loops=426)
    -> Filter: ((w.dailyAvg < 45) and ((t1.type = 'jacket') or ((t1.type = 'long sleeve') or (ifnull(t2.type,'') = 'long sleeve')))) or ((w.dailyAvg > 45) and ((t1.type = 'jacket') or ((t1.type = 'long sleeve') or (ifnull(t2.type,'') = 'long sleeve')))) or ((w.dailyAvg < 35) or ((t1.type = 'jacket') or ((t1.type = 'long sleeve') or (ifnull(t2.type,'') = 'long sleeve'))))
      -> Nested loop left join (cost=22.3 rows=2.68) (actual time=0.0194..0.033 rows=2..3 loops=426)
        -> Nested loop inner join (cost=21.4 rows=2.68) (actual time=0.0173..0.032 rows=2..3 loops=426)
          -> Nested loop inner join (<hash>(t1.userId)<hash>(t1.location)) (cost=21.4 rows=2.68) (actual time=0.0173..0.032 rows=2..3 loops=426)
            -> Filter: ((t1.userId = t2.userId) and (t1.location <= t2.location)) (cost=2.64 rows=20) (actual time=0.00584..0.00848 rows=9.59 loops=426)
              -> Covering index lookup on b using idx userid_userid (cost=2.64 rows=20) (actual time=0.00544..0.00654 rows=9.59 loops=426)
                -> Filter: (o2.toplayer1 is not null) (cost=0.342 rows=34) (actual time=0.00164..0.00172 rows=0..24 loops=4084)
                -> Index lookup on o using bottom (bottom=o.clothingId) (cost=0.342 rows=34) (actual time=0.00164..0.00172 rows=0..24 loops=4084)
                  -> Filter: ((w.dailyAvg < 35) or ((w.dailyAvg > 45) and (t1.userId = t2.userId))) (cost=0.25 rows=0..1) (actual time=0.00143..0.0015 rows=1 loops=1)
                    -> Single-row index lookup on t1 using PRIMARY (clothingId=o2.topLayer1) (cost=0.25 rows=1) (actual time=0.00109..0.00111 rows=1 loops=979)
                    -> Filter: (t2.userId = t1.userId) (cost=0.287 rows=1) (actual time=979e-6..6.00102 rows=0..728 loops=979)
                      -> Single-row index lookup on t2 using PRIMARY (clothingId=o2.toplayer2) (cost=0.287 rows=1) (actual time=806e-6..818e-6 rows=0..728 loops=979)

s=979)

```

	Cost
Default	1226
idx_loc_date	162
idx_outfitid	1361
idx_userid_type	1367

We put indexes on the weather, clothingItems, and outfits table(join clauses). The only beneficial one was the index on the weather join. This significantly decreased the cost, so this was our best outcome.

### QUERY #4

```

SELECT u.userId, u.location, o.outfitId, t1.name AS topLayer1_name, t2.name AS topLayer2_name, b.name AS bottom_name, r.name AS restaurant_name, r.rest_type
FROM users u
JOIN outfits o ON o.outfitId = (
    SELECT o2.outfitId
    FROM outfits o2
    JOIN clothingItems t1 ON o2.topLayer1 = t1.clothingId AND t1.userId = u.userId
    LEFT JOIN clothingItems t2 ON o2.topLayer2 = t2.clothingId AND t2.userId = u.userId
    JOIN clothingItems b ON o2.bottom = b.clothingId AND b.userId = u.userId
    WHERE t1.style = 'casual' AND b.style = 'casual'
    LIMIT 1
)
JOIN clothingItems t1 ON o.topLayer1 = t1.clothingId AND t1.userId = u.userId
LEFT JOIN clothingItems t2 ON o.topLayer2 = t2.clothingId AND t2.userId = u.userId
JOIN clothingItems b ON o.bottom = b.clothingId AND b.userId = u.userId
JOIN restaurants r ON r.location = u.location
WHERE (r.rest_type LIKE '%Casual%' OR r.rest_type LIKE '%Quick Bites%')
AND r.url = (
    SELECT r2.url
    FROM restaurants r2
    WHERE r2.location = u.location
    AND (r2.rest_type LIKE '%Casual%' OR r2.rest_type LIKE '%Quick Bites%')
    LIMIT 1
);

```

This is another advanced query that suggests an outfit for a user, at their location, which matches the style to the restaurant. This query will also be a key component to recommending restaurants. It can be further changed to select all sorts of styles and types of restaurants but this is just an example with casual mapping to quick bites or casual dining restaurants.

```

mysql> SELECT u.userId, u.location, o.outfitId, t1.name AS topLayer1_name, t2.name AS topLayer2_name, b.name AS bottom_name, r.name AS restaurant_name, r.rest_type
FROM users u JOIN outfits o ON o.outfitId = (
    SELECT o2.outfitId
    FROM outfits o2
    JOIN clothingItems t1 ON o2.topLayer1 = t1.clothingId AND t1.userId = u.userId
    LEFT JOIN clothingItems t2 ON o2.topLayer2 = t2.clothingId AND t2.userId = u.userId
    WHERE t1.style = 'casual' AND b.style = 'casual'
    LIMIT 1
)
JOIN clothingItems t1 ON o.topLayer1 = t1.clothingId AND t1.userId = u.userId
LEFT JOIN clothingItems t2 ON o.topLayer2 = t2.clothingId AND t2.userId = u.userId
JOIN clothingItems b ON o.bottom = b.clothingId AND b.userId = u.userId
JOIN restaurants r ON r.location = u.location
WHERE (r.rest_type LIKE '%Casual%' OR r2.rest_type LIKE '%Quick Bites%')
AND r.url = (
    SELECT r2.url
    FROM restaurants r2
    WHERE r2.location = u.location
    AND (r2.rest_type LIKE '%Casual%' OR r2.rest_type LIKE '%Quick Bites%')
    LIMIT 1
)
LIMIT 15;
+-----+-----+-----+-----+-----+-----+-----+-----+
| userId | location | outfitId | topLayer1_name | topLayer2_name | bottom_name | restaurant_name | rest_type |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | New York, NY | 7002 | red long sleeve | green jacket | yellow jeans | Jalas | Casual Dining |
| 2 | Dallas, TX | 7007 | brown t-shirt | black pants | black pants | Quick Bites | Casual Dining |
| 3 | San Francisco, CA | 7010 | green long sleeve | beige rain coat | white pants | Tejas Bar and Restaurant | Casual Dining |
| 4 | San Francisco, CA | 7013 | navy t-shirt | NULL | yellow jeans | Tejas Bar and Restaurant | Casual Dining |
| 5 | Austin, TX | 7019 | blue long sleeve | brown jacket | navy jeans | Pizza Palace | Quick Bites |
| 6 | Las Vegas, NV | 7022 | blue long sleeve | black rain coat | black pants | Kogi Korean BBQ | Casual Dining |
| 7 | Seattle, WA | 7030 | green t-shirt | yellow rain coat | blue pants | Donne Biriyani House | Quick Bites |
| 8 | Los Angeles, CA | 7038 | blue long sleeve | NULL | blue pants | Barbecue Nation | Casual Dining |
| 9 | Phoenix, AZ | 7040 | brown t-shirt | red rain coat | beige pants | Curry with a 'K' - St. Mark's Hotel | Casual Dining |
| 10 | Los Angeles, CA | 7043 | blue long sleeve | black rain coat | black pants | Hakkasan | Casual Dining |
| 11 | Denver, CO | 7049 | gray t-shirt | white rain coat | white jeans | Kerala Pavilion | Quick Bites |
| 12 | Dallas, TX | 7053 | brown long sleeve | green jacket | blue pants | Beijing Bites | Casual Dining |
| 13 | Minneapolis, MN | 7057 | navy t-shirt | NULL | gray jeans | Roti Shotti | Quick Bites |
| 14 | Detroit, MI | 7066 | green long sleeve | white rain coat | navy jeans | Kerala Pavilion | Quick Bites |
| 15 | Houston, TX | 7069 | beige t-shirt | blue jacket | red jeans | Ebony | Casual Dining |
+-----+-----+-----+-----+-----+-----+-----+-----+
15 rows in set (1.89 sec)

```

```

| -> Nested loop inner join (cost=459 rows=139) (actual_time=10..4212 rows=74 loops=1)
    -> Nested loop left join (cost=82.1 rows=0.25) (actual_time=1..130 rows=74 loops=1)
        -> Nested loop inner join (cost=80.2 rows=5) (actual_time=1.29..129 rows=74 loops=1)
            -> Nested loop inner join (cost=45.2 rows=100) (actual_time=1..28..128 rows=74 loops=1)
                -> Table scan on u (cost=10.2 rows=100) (actual_time=0.267..0.684 rows=100 loops=1)
                -> Filter: ((o.outfitid <= 100) and (o.toplayer1 is not null) and (o.bottom is not null)) (cost=0.251 rows=1) (actual_time=1.27..1.27 rows=0.74 loops=100)
                -> Select #2 (subquery in condition; dependent)
                    -> Limit: 1 row(s) (cost=7..12 rows=0.134) (actual_time=0..506..0..506 rows=0.895 loops=248)
                    -> Nested loop left join (cost=45.2 rows=100) (actual_time=0.239..0.235 rows=100 loops=1)
                        -> Table scan on u (cost=10.2 rows=100) (actual_time=0.231..0.233 rows=100 loops=1)
                        -> Filter: ((o.outfitid <= 100) and (o.toplayer1 is not null) and (o.bottom is not null)) (cost=0.251 rows=1) (actual_time=0.231..0.233 rows=0.74 loops=100)
                        -> Nested loop inner join (cost=6..14 rows=2.68) (actual_time=0..493..0..497 rows=1..58 loops=248)
                            -> Filter: (b.style = 'casual') (cost=5..2 rows=2)
                                -> Index lookup on b using idx_clothing_style (cost=0..11 rows=2) (actual_time=0.045..0.046 rows=2 loops=248)
                            -> Nested loop inner join (cost=6..14 rows=2.68) (actual_time=0..493..0..497 rows=1..58 loops=248)
                                -> Filter: ((t1.style = 'casual') and (t1.userid = u.userid)) (cost=0.252 rows=0.05) (actual_time=0.00315..0..00315 rows=0.568 loops=391)
                            -> Filter: (t2.userid = u.userid) (cost=0.251 rows=1) (actual_time=0.00242..0..00242 rows=0.689 loops=222)
                                -> Single-row index lookup on t2 using PRIMARY (clothingId=o2.toplayer2) (cost=0..996 rows=1) (actual_time=0..00207..0..00207 rows=0.689 loops=222)
                            -> Filter: (t1.userid = u.userid) (cost=0..25 rows=0.05) (actual_time=0.00516..0..00516 rows=1 loops=74)
                                -> Single-row index lookup on t1 using PRIMARY (clothingId=o2.toplayer1) (cost=0..25 rows=1) (actual_time=0..00458..0..00463 rows=1 loops=74)
                            -> Filter: (t2.userid = u.userid) (cost=0..65 rows=1) (actual_time=0.0389..0..0447 rows=1 loops=74)
                                -> Single-row index lookup on b using PRIMARY (clothingId=o2.bottom) (cost=0..251 rows=1) (actual_time=0.00442..0..00452 rows=1 loops=74)
                            -> Filter: (((t1.style = 'casual') or (t2.style = 'casual')) and (t1.location = u.location)) (cost=0..1464 rows=555) (actual_time=3..28..55.2 rows=1 loops=74)
                                -> Index lookup on r using idx_restaurants_location (location=u.location), with index condition: (r.location = u.location) (cost=0..1464 rows=546) (actual_time=3..14..39.6 rows=2639 loops=74)
                            -> Select #3 (subquery in condition; dependent)
                                -> Limit: 1 row(s) (cost=1297 rows=1) (actual_time=0..00567..0..00606 rows=1 loops=129345)
                                -> Filter: (((t2.rest_type like '%Casual%' or (t2.rest_type like '%Quick Bites$%')) and (t2.location = u.location)) (cost=1297 rows=555) (actual_time=0..00663..0..00663 rows=1 loops=129345)
                                    -> Index lookup on r2 using idx_restaurants_location (location=u.location), with index condition: (r2.location = u.location) (cost=1297 rows=2646) (actual_time=0..00376..0..00566 rows=2 loops=129345)

```

## Index #1:

CREATE INDEX idx\_style ON clothingItems(userId, style);  
DROP INDEX idx\_style ON clothingItems;

```

-----+
| -> Nested loop inner join (cost=459 rows=139) (actual_time=55..3819 rows=74 loops=1)
    -> Nested loop left join (cost=82.1 rows=0.25) (actual_time=55..114 rows=74 loops=1)
        -> Nested loop inner join (cost=80.2 rows=5) (actual_time=55..113 rows=74 loops=1)
            -> Nested loop inner join (cost=45.2 rows=100) (actual_time=55..113 rows=74 loops=1)
                -> Table scan on u (cost=10.2 rows=100) (actual_time=0..587..0..471 rows=100 loops=1)
                -> Filter: ((o.outfitid <= 100) and (o.toplayer1 is not null) and (o.bottom is not null)) (cost=0.251 rows=1) (actual_time=1..12..1..12 rows=0.74 loops=100)
                -> Select #2 (subquery in condition; dependent)
                    -> Limit: 1 row(s) (cost=5..55 rows=0.299) (actual_time=0..444..0..444 rows=0.895 loops=248)
                    -> Nested loop inner join (cost=5..55 rows=0.299) (actual_time=0..443..0..443 rows=0.895 loops=248)
                        -> Nested loop inner join (cost=3..35 rows=5..97) (actual_time=0..273..0..278 rows=1..58 loops=248)
                            -> Covering index lookup on b using idx_style (userId=u.userid, style='casual') (cost=1..26 rows=4..45) (actual_time=0..0344..0..0364 rows=5..65 loops=248)
                            -> Filter: (b.toplayer1 is not null) (cost=0..365 rows=1..34) (actual_time=0..0425..0..0426 rows=0..279 loops=1400)
                            -> Nested loop inner join (cost=2..25 rows=0.05) (actual_time=0..043..0..043 rows=0..051 loops=391)
                            -> Filter: ((t1.style = 'casual') and (t1.userid = u.userid)) (cost=0..251 rows=0..05) (actual_time=0..099..0..099 rows=0..0568 loops=391)
                            -> Single-row index lookup on t1 using PRIMARY (clothingId=o2.toplayer1) (cost=0..251 rows=1) (actual_time=0..0977..0..0977 rows=1 loops=391)
                            -> Filter: (t2.userid = u.userid) (cost=0..25 rows=0..05) (actual_time=0..0072..0..0073 rows=1 loops=74)
                                -> Single-row index lookup on t1 using PRIMARY (clothingId=o2.toplayer1) (cost=0..251 rows=1) (actual_time=0..00653..0..00658 rows=1 loops=74)
                            -> Filter: (t1.userid = u.userid) (cost=0..251 rows=0..05) (actual_time=0..00679..0..00679 rows=1 loops=74)
                                -> Single-row index lookup on b using PRIMARY (clothingId=o2.bottom) (cost=0..251 rows=1) (actual_time=0..00679..0..00679 rows=1 loops=74)
                            -> Filter: ((t2.userid = u.userid) or (t2.rest_type like '%Casual%' or (t2.rest_type like '%Quick Bites$%')) and (t2.location = u.location)) (cost=1464 rows=555) (actual_time=2..1..50.1 rows=1 loops=74)
                                -> Index lookup on r using idx_restaurants_location (location=u.location), with index condition: (r.location = u.location) (cost=1464 rows=2646) (actual_time=2..04..34.6 rows=2639 loops=74)
                            -> Select #3 (subquery in condition; dependent)
                                -> Limit: 1 row(s) (cost=1297 rows=1) (actual_time=0..00679..0..00682 rows=1 loops=129345)
                                -> Filter: (((t2.rest_type like '%Casual%' or (t2.rest_type like '%Quick Bites$%')) and (t2.location = u.location)) (cost=1297 rows=555) (actual_time=0..00665..0..00665 rows=1 loops=129345)
                                    -> Index lookup on r2 using idx_restaurants_location (location=u.location), with index condition: (r2.location = u.location) (cost=1297 rows=2646) (actual_time=0..00372..0..0057 rows=2 loops=129345)

```

## Index #2:

CREATE INDEX idx\_restaurants\_type ON restaurants(location, rest\_type);  
DROP INDEX idx\_restaurants\_type ON restaurants;

```

-----+
| -> Nested loop inner join (cost=531 rows=139) (actual_time=36..3865 rows=74 loops=1)
    -> Nested loop left join (cost=144 rows=0.25) (actual_time=206..295 rows=74 loops=1)
        -> Nested loop inner join (cost=144 rows=0.25) (actual_time=236..285 rows=74 loops=1)
            -> Nested loop inner join (cost=144 rows=5) (actual_time=236..284 rows=74 loops=1)
                -> Table scan on r (cost=10.2 rows=5) (actual_time=4..74..4 rows=5 loops=1)
                -> Filter: ((o.outfitid = select #2) and (o.toplayer1 is not null) and (o.bottom is not null)) (cost=0.251 rows=1) (actual_time=2..37..2..37 rows=0.74 loops=100)
                -> Single-row index lookup on o using PRIMARY (outfitid=select #2) (cost=0.251 rows=1) (actual_time=2..33..2..33 rows=0.74 loops=100)
                    -> Select #2 (subquery in condition; dependent)
                        -> Limit: 1 row(s) (cost=0..949 rows=0..895 loops=248)
                        -> Nested loop left join (cost=46 rows=100) (actual_time=0..948..0..948 rows=0..895 loops=248)
                            -> Nested loop inner join (cost=20..4 rows=0.134) (actual_time=0..948..0..948 rows=0..895 loops=248)
                                -> Nested loop inner join (cost=20..3 rows=0.134) (actual_time=0..946..0..946 rows=0..895 loops=248)
                                    -> Filter: ((t1.style = 'casual') and (t1.userid = u.userid)) (cost=0..838 rows=0..09) (actual_time=0..0309..0..0309 rows=0..568 loops=391)
                                    -> Nested loop inner join (cost=17..8 rows=2..68) (actual_time=0..936..0..94 rows=1..58 loops=248)
                                        -> Filter: ((t2.style = 'casual') and (t2.location = u.location)) (cost=0..838 rows=0..09) (actual_time=0..0309..0..0309 rows=0..568 loops=391)
                                        -> Filter: (t2.userid = u.userid) (cost=1..58 rows=1) (actual_time=0..00226..0..00226 rows=0..689 loops=222)
                                            -> Single-row index lookup on t2 using PRIMARY (clothingId=o2.toplayer2) (cost=1..58 rows=1) (actual_time=0..00202..0..00202 rows=0..689 loops=222)
                                            -> Filter: (t1.userid = u.userid) (cost=0..833 rows=0..05) (actual_time=0..00516..0..00532 rows=1 loops=74)
                                                -> Single-row index lookup on r using PRIMARY (location=u.location) (cost=0..833 rows=1) (actual_time=0..00439..0..00442 rows=1 loops=74)
                                            -> Filter: (t2.userid = u.userid) (cost=0..834 rows=0..05) (actual_time=0..0042..0..00439 rows=1 loops=74)
                                                -> Single-row index lookup on b using PRIMARY (clothingId=o2.bottom) (cost=0..834 rows=1) (actual_time=0..00373..0..00383 rows=1 loops=74)
                                            -> Filter: (((t1.userid = u.userid) or (t1.location = u.location)) and (t1.location = u.location)) (cost=1464 rows=555) (actual_time=0..00343..0..00359 rows=0..689 loops=74)
                                                -> Index lookup on r using idx_restaurants_location (location=u.location), with index condition: (r.location = u.location) (cost=1464 rows=2646) (actual_time=0..99..45.7 rows=1 loops=74)
                                            -> Select #3 (subquery in condition; dependent)
                                                -> Limit: 1 row(s) (cost=1297 rows=1) (actual_time=0..00664..0..00664 rows=1 loops=129345)
                                                -> Filter: (((t2.rest_type like '%Casual%' or (t2.rest_type like '%Quick Bites$%')) and (t2.location = u.location)) (cost=1297 rows=555) (actual_time=0..00627..0..00627 rows=1 loops=129345)
                                                    -> Index lookup on r2 using idx_restaurants_location (location=u.location), with index condition: (r2.location = u.location) (cost=1297 rows=2646) (actual_time=0..00346..0..00534 rows=2 loops=129345)

```

## Index #3:

CREATE INDEX idx\_url\_filter ON restaurants(location, rest\_type, url);  
DROP INDEX idx\_url\_filter ON restaurants;

```

| -> Nested loop inner join (cost=567 rows=139) (actual time=1255..38687 rows=74 loops=1)
  -> Nested loop left join (cost=191 rows=0.25) (actual time=183..725 rows=74 loops=1)
    -> Nested loop inner join (cost=183 rows=1) (actual time=183..1 loops=1)
      -> Nested loop inner join (cost=183 rows=5) (actual time=183..724 rows=74 loops=1)
        -> Nested loop inner join (cost=83.5 rows=100) (actual time=182..724 rows=74 loops=1)
          -> Table scan on o (cost=1.0 rows=100) (actual time=182..141 rows=100 loops=1)
            -> Filter: ((o.outfitId = t2.outfitId) AND (o.outfitStyle is not null) AND (o.bottom is not null)) (cost=0.626 rows=1) (actual time=5.82..5.82 rows=0.74 loops=100)
              -> Single-row index lookup on o using PRIMARY (outfitId=selected #2) (cost=0.626 rows=1) (actual time=5.78..5.78 rows=0.74 loops=100)
                -> Select #2 (subquery in condition; dependent)
                  -> Limit: 1 row(s) (cost=23.3 rows=0.134) (actual time=2.34..2.34 rows=0.895 loops=248)
                    -> Nested loop left join (cost=23.2 rows=0.134) (actual time=2.34..2.34 rows=0.895 loops=248)
                      -> Nested loop inner join (cost=20.5 rows=2.68) (actual time=2.32..2.33 rows=1.58 loops=248)
                        -> Nested loop inner join (cost=18.5 rows=2) (actual time=1.87..1.87 rows=1.69 loops=248)
                          -> Filter: (b.style = 'casual') (cost=18.5 rows=2) (actual time=1.87..1.87 rows=9.11 loops=248)
                            -> Filter: ((t1.topPlayerId is not null)) (cost=0.905 rows=1.34) (actual time=0.0806..0.0808 rows=0.279 loops=1400)
                              -> Index lookup on o2 using bottom (bottom=b.clothingId) (cost=0.905 rows=1.34) (actual time=0.0804..0.0805 rows=0.279 loops=1400)
                            -> Filter: ((t1.style = 'casual') AND (t1.userId = u.userId)) (cost=0.919 rows=0.05) (actual time=0.00468..0.00468 rows=0.568 loops=391)
                              -> Index lookup on r using idx_restaurants_type (rootId=r.rootId) (cost=0.919 rows=1) (actual time=0.00468..0.00468 rows=1 loops=391)
                            -> Filter: (t2.userId = u.userId) (cost=1..66 rows=1) (actual time=0.00468..0.00468 rows=1 loops=222)
                              -> Single-row index lookup on t2 using PRIMARY (clothingId=o2.topPlayerId) (cost=1..66 rows=1) (actual time=0.00427..0.00427 rows=0.689 loops=222)
                                -> Filter: (t1.userId = t2.userId) (cost=0.917 rows=0.05) (actual time=0.0056..0.00597 rows=1 loops=74)
                                  -> Single-row index lookup on r using PRIMARY (clothingId=t2.rootId) (cost=0.917 rows=1) (actual time=0.00533 rows=1 loops=74)
                                    -> Filter: ((t2.userId = u.userId) AND (t2.rootId = r.rootId)) (cost=0.918 rows=1) (actual time=0.00491..0.00502 rows=1 loops=74)
                                  -> Single-row index lookup on b using PRIMARY (clothingId=o.bottom) (cost=0.918 rows=1) (actual time=0.00491..0.00502 rows=1 loops=74)
                                -> Filter: (t2.userId = u.userId) (cost=1..32 rows=1) (actual time=0.00578..0.00647 rows=0.689 loops=74)
                                  -> Index lookup on r using idx_restaurants_location (location=r.location) (cost=0.6529..0.6905 rows=0.689 loops=74)
                                    -> Filter: (((r.rootId like '%Quick Bites%') OR (r.rootId like '%Casual%')) AND ((r2.rootType like '%Quick Bites%') OR (r2.rootType like '%Casual%'))) (cost=1444 rows=555) (actual time=241..513 rows=1 loops=74)
                                      -> Filter: ((r.rootId like '%Quick Bites%') OR (r.rootId like '%Casual%')) (cost=1444 rows=555) (actual time=241..513 rows=1 loops=74)
                                        -> Index lookup on r using idx_url_filter (location=u.location) (cost=1444 rows=2646) (actual time=5..35..128 rows=2639 loops=74)
                                          -> Select #3 (subquery in condition; dependent)
                                            -> Limit: 1 row(s) (cost=525 rows=1) (actual time=0.216..0.216 rows=1 loops=129345)
                                              -> Filter: (((r2.rootType like '%Quick Bites%') OR (r2.rootType like '%Casual%')) AND ((r2.rootType like '%Quick Bites%') OR (r2.rootType like '%Casual%'))) (cost=525 rows=586) (actual time=0.217..0.217 rows=1 loops=129345)
                                                -> Covering index lookup on r2 using idx_url_filter (location=u.location) (cost=525 rows=2793) (actual time=0.012..0.136 rows=130 loops=129345)

```

	Cost
Default	459
idx_style	459
idx_restaurants_type	521
idx_url_filter	567

We put indexes on the clothingItems and restaurants tables(join clauses and group by). We saw an increase in cost for most and no change for the clothingItems index. We ultimately decided that the default index was our best choice.