DDL Commands

```
CREATE TABLE WorldCities(
    city VARCHAR(50) NOT NULL,
    city_ascii VARCHAR(50) NOT NULL,
    lat DECIMAL(13, 10) NOT NULL,
    lng DECIMAL(13, 10) NOT NULL,
    country VARCHAR(50)
    iso2 VARCHAR(4),
    iso3 VARCHAR(4),
    admin_name VARCHAR(50) NOT NULL,
    capital VARCHAR(50) NOT NULL,
    population INT NOT NULL,
    id VARCHAR(50)
    PRIMARY KEY(id)
);

CREATE TABLE VacationSpots(
    VacationSpotName VARCHAR(50),
    CityId INT NOT NULL,
    LikeCount INT,
    PRIMARY KEY (VacationSpotName),
    FOREIGN KEY (CityId) REFERENCES Cities(CityId) ON DELETE CASCADE
);

CREATE TABLE UserAccounts(
    Username VARCHAR(50),
    UserPassword VARCHAR(100) NOT NULL,
    ProfilePictureUrl VARCHAR(255),
    ProfileDescription VARCHAR(255),
    Gender VARCHAR(20),
    Country VARCHAR(50),
    Age INT,
    PRIMARY KEY (Username)
);

CREATE TABLE Reviews(
    ReviewID INT,
    Username VARCHAR(50) NOT NULL,
    ReviewText VARCHAR(2000),
    ReviewRating INT,
    CreatedAt DATETIME,
    UpdatedAt DATETIME,
    LikeCount INT,
    PRIMARY KEY (ReviewID),
```

```
    FOREIGN KEY (Username) REFERENCES UserAccounts(Username) ON DELETE
        CASCADE
);

CREATE TABLE Images(
    ImageURL VARCHAR(255),
    ReviewID INT NOT NULL,
    PRIMARY KEY (ImageURL),
    FOREIGN KEY (ReviewID) REFERENCES Reviews(ReviewID) ON DELETE CASCADE,
);

CREATE TABLE Follows(
    followerUsername INT,
    followeeUsername INT,
    PRIMARY KEY (followerUsername, followeeUsername),
    FOREIGN KEY (followerUsername) REFERENCES UserAccounts(Username) ON DELETE
        CASCADE,
    FOREIGN KEY (followeeUsername) REFERENCES UserAccounts(Username) ON DELETE
        CASCADE
);

CREATE TABLE FavoriteSpots(
    Username VARCHAR(50),
    VacationSpotName VARCHAR(50),
    PRIMARY KEY(Username, VacationSpotName),
    FOREIGN KEY (Username) REFERENCES UserAccounts(Username) ON DELETE
        CASCADE,
    FOREIGN KEY (VacationSpotName) REFERENCES VacationSpots(VacationSpotName) ON
        DELETE CASCADE
);

CREATE TABLE VacationSpotReviews(
    ReviewId INT NOT NULL,
    VacationSpotName VARCHAR(50),
    PRIMARY KEY (ReviewId, VacationSpotName),
    FOREIGN KEY (ReviewId) REFERENCES Reviews(ReviewId) ON DELETE CASCADE,
    FOREIGN KEY (VacationSpotName) REFERENCES VacationSpots(VacationSpotName) ON
        DELETE CASCADE
);
```

Query Results (Located in db/queries.sql)

Query #1 - Get recent reviews from user's following and from the top 20 followed users

```
mysql> SELECT ReviewID, Username, ReviewText, ReviewRating, CreatedAt, LikeCount
    -> FROM (
    ->     SELECT r.ReviewID, r.Username, r.ReviewText, r.ReviewRating, r.CreatedAt, r.LikeCount
    ->     FROM Reviews r
    ->     JOIN Follows f ON r.Username = f.followeeUsername
    ->     WHERE f.followerUsername = 'nancy57'
    ->
    ->     UNION
    ->
    ->     SELECT r.ReviewID, r.Username, r.ReviewText, r.ReviewRating, r.CreatedAt, r.LikeCount
    ->     FROM Reviews r
    ->     JOIN (
    ->         SELECT f.followeeUsername
    ->         FROM Follows f
    ->         GROUP BY f.followeeUsername
    ->         ORDER BY COUNT(DISTINCT f.followerUsername) DESC
    ->         LIMIT 20
    ->     ) AS TopUsers
    ->     ON r.Username = TopUsers.followeeUsername
    -> ) AS ReviewFeed
    -> ORDER BY CreatedAt DESC
    -> LIMIT 15;
```

ReviewID	Username	ReviewText	ReviewRating	CreatedAt	LikeCount
565	kimberly00	His argue including value decision sometimes particularly. The destination was amazing. The relaxing was amazing. The ocean was amazing. The getaway was amazing.	5	2025-02-04 10:15:36	5
977	barbararodriguez	Power network indicate subject our head born suddenly. The mountain was amazing. The destination was amazing. The tranquil was amazing. The wildlife was amazing.	4	2024-11-26 06:04:20	193
638	kimberly00	Know concern industry join. The sunset was amazing. The resort was amazing. The beach was amazing. The escape was amazing.	1	2024-11-10 08:08:18	114
438	thomas28	Fly class outside among PM forget. The luxury was amazing. The tranquil was amazing. The beach was amazing.	1	2024-10-10 17:25:27	864
713	amanda32	Poor environment job. The sunset was amazing. The scenic was amazing. The wildlife was amazing. The escape was amazing.	4	2024-08-23 05:56:21	387
726	armstrongsara	Involve instead just ball. The cabin was amazing. The luxury was amazing. The destination was amazing.	2	2024-08-23 02:51:23	447
637	amanda32	Direction none bad type member explain. The luxury was amazing. The wildlife was amazing. The pool was amazing. The scenic was amazing.	2	2024-08-16 00:49:47	39
527	john80	Area PM someone assume social war generation. The relaxing was amazing. The spa was amazing. The island was amazing. The resort was amazing.	3	2024-08-01 07:25:56	860
265	thomas28	Condition manage admit maintain bit. The adventure was amazing. The island was amazing. The tranquil was amazing.	3	2024-06-30 20:30:18	126
758	ramseyvalerie	Agency teacher shake individual executive born investment. The escape was amazing. The relaxing was amazing. The luxury was amazing.	2	2024-05-23 09:53:06	956
853	kimberly00	Measure minute address writer you to. The paradise was amazing. The tranquil was amazing.	1	2024-05-03 18:31:47	815
738	michelle96	Vote tree oil understand policy can bag try. The destination was amazing. The pool was amazing.	1	2024-03-16 07:24:55	811
283	ramseyvalerie	Coach maybe energy million build charge arrive loss. The spa was amazing. The beach was amazing. The ocean was amazing. The paradise was amazing.	1	2024-02-16 14:41:54	525
149	abbottchristopher	With war none company recent city large. The cabin was amazing. The island was amazing. The paradise was amazing.	1	2024-02-06 21:47:10	46
600	barbararodriguez	Smile front test southern middle reach late. The island was amazing. The resort was amazing. The mountain was amazing.	5	2024-02-02 09:49:14	500

15 rows in set (0.05 sec)

Query #2: Top 3 reviews for a given vacation spot that have a like count greater than or equal to the average like count of all reviews for that vacation spot. There are not 15 records because we only want to display the top 3

```
Database changed
mysql> SELECT
    ->     r.ReviewID,
    ->     r.Username,
    ->     r.ReviewText,
    ->     r.ReviewRating,
    ->     r.CreatedAt,
    ->     r.LikeCount,
    ->     vsr.VacationSpotName,
    ->     i.ImageURL
    -> FROM Reviews r
    -> JOIN VacationSpotReviews vsr ON r.ReviewID = vsr.ReviewID
    -> LEFT JOIN Images i ON r.ReviewID = i.ReviewID
    -> WHERE vsr.VacationSpotName =  'Santana-Flowers Resort'
    -> AND r.LikeCount >= (
    ->     SELECT AVG(LikeCount) FROM Reviews
    ->     WHERE ReviewID IN (
    ->         SELECT ReviewID FROM VacationSpotReviews
    ->         WHERE VacationSpotName =  'Santana-Flowers Resort'
    ->     )
    -> )
    -> ORDER BY r.LikeCount DESC, r.CreatedAt DESC
    -> LIMIT 3;
```

ReviewID	Username	ReviewText	ReviewRating	CreatedAt	LikeCount	VacationSpotName	ImageURL
305	xcox	Source against wear such leg discover catch. The resort was amazing. The paradise was amazing. The island was amazing.	3	2024-08-03 11:06:08	997	Santana-Flowers Resort	NULL
137	rwilcox	Late six national into address heart budget. The wildlife was amazing. The ocean was amazing. The mountain was amazing.	5	2024-08-22 07:54:40	994	Santana-Flowers Resort	NULL
722	vdavis	Second even agency south account form his. The cabin was amazing. The ocean was amazing. The beach was amazing.	4	2024-12-14 11:30:05	990	Santana-Flowers Resort	NULL

Query #3: Find the most relevant vacation spots based on its popularity determined by number of reviews and average rating

VacationSpotName	city_ascii	TotalReviews	AverageRating
Pacaya Volcano	Jalalabad	1345	2.1375
Antigua Guatemala	Jalalabad	784	3.5689
Tikal National Park	Jalalabad	661	4.5673
Lake Atitlán	Jalalabad	554	4.1552
Semuc Champey	Jalalabad	275	2.4073
Santana-Flowers Resort	Jalalabad	5	3.8000

Query #4: Gets users favorite vacation spots that are also popular vacation spots (have greater than the average amount of likes of vacation spots)

```
mysql> SELECT fs.Username, fs.VacationSpotName, c.city, v.LikeCount
    -> FROM FavoriteSpots fs
    -> JOIN VacationSpots v ON fs.VacationSpotName = v.VacationSpotName
    -> JOIN WorldCities c ON v.CityId = c.id
    -> WHERE fs.Username = 'aaronjones'
    ->
    -> INTERSECT
    ->
    -> SELECT fs.Username, fs.VacationSpotName, c.city, v.LikeCount
    -> FROM FavoriteSpots fs
    -> JOIN VacationSpots v ON fs.VacationSpotName = v.VacationSpotName
    -> JOIN WorldCities c ON v.CityId = c.id
    -> WHERE v.LikeCount >= (
    ->    SELECT AVG(LikeCount) FROM VacationSpots
    -> )
    -> ORDER BY LikeCount DESC
    -> LIMIT 15;
+------------+-----------------------------------+------------------+-----------+
| Username   | VacationSpotName                  | city             | LikeCount |
+------------+-----------------------------------+------------------+-----------+
| aaronjones | Walls-Hayden Resort               | Ierápetra        |      4979 |
| aaronjones | Fowler-Arias Resort               | Mirbāṭ           |      4974 |
| aaronjones | Hill Inc Resort                   | Oer-Erkenschwick |      4968 |
| aaronjones | Long LLC Resort                   | Falköping        |      4967 |
| aaronjones | Manning, Marshall and Stevens Resort | Hamilton      |      4954 |
| aaronjones | Duran Inc Resort                  | Malatya          |      4949 |
| aaronjones | Gibson, Wilson and Wagner Resort  | Argayash         |      4946 |
| aaronjones | Acosta-Phillips Resort            | Mohale's Hoek    |      4922 |
| aaronjones | Schroeder Inc Resort              | Majayjay         |      4902 |
| aaronjones | Ellis-Valdez Resort               | Two Rivers       |      4900 |
| aaronjones | Turner, Ramirez and Harris Resort | Ban Si Don Chai  |      4897 |
| aaronjones | Howell Group Resort               | Gurh             |      4890 |
| aaronjones | Anderson Group Resort             | Phillipsburg     |      4889 |
| aaronjones | Garcia, Torres and Rosario Resort | Anan             |      4882 |
| aaronjones | Mcclain Group Resort              | Tangcun          |      4860 |
+------------+-----------------------------------+------------------+-----------+
15 rows in set (0.02 sec)
```

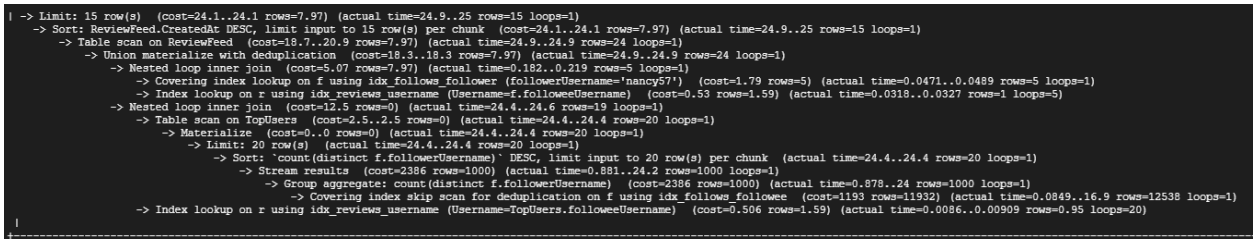
Column Counts

```
mysql> SELECT table_name, table_rows
    -> FROM information_schema.tables
    -> WHERE table_schema = DATABASE();
+---------------------+------------+
| TABLE_NAME          | TABLE_ROWS |
+---------------------+------------+
| CostOfLiving        |       5186 |
| FavoriteSpots       |        546 |
| Follows             |      11931 |
| Images              |      83599 |
| Reviews             |       1001 |
| Temperatures        |       9690 |
| UserAccounts        |       1001 |
| VacationSpotReviews |         20 |
| VacationSpots       |        532 |
| WorldCities         |      23424 |
+---------------------+------------+
10 rows in set (0.15 sec)
```

Indexing

-- Query 1

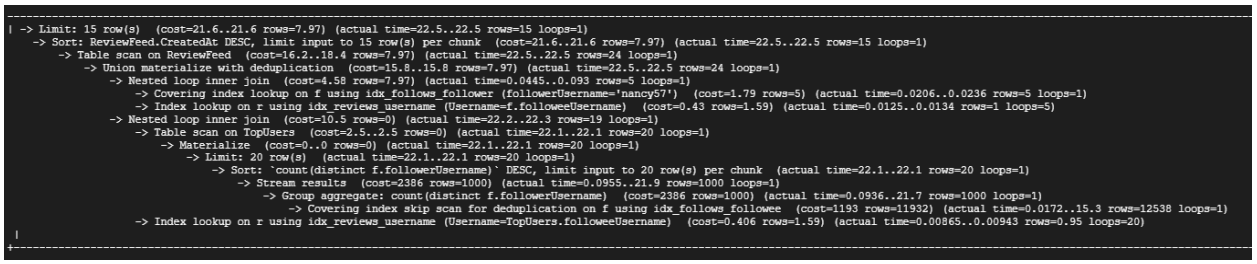
Before Indexing



Cost = 24.1

Design 1

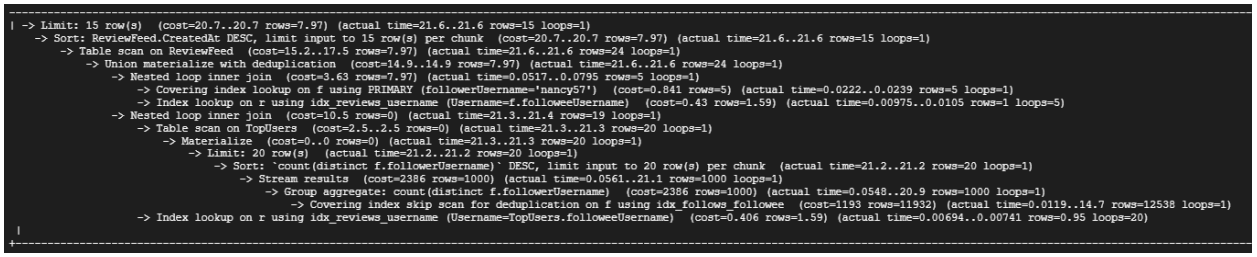
CREATE INDEX rating ON Reviews(ReviewRating);



Cost = 21.6

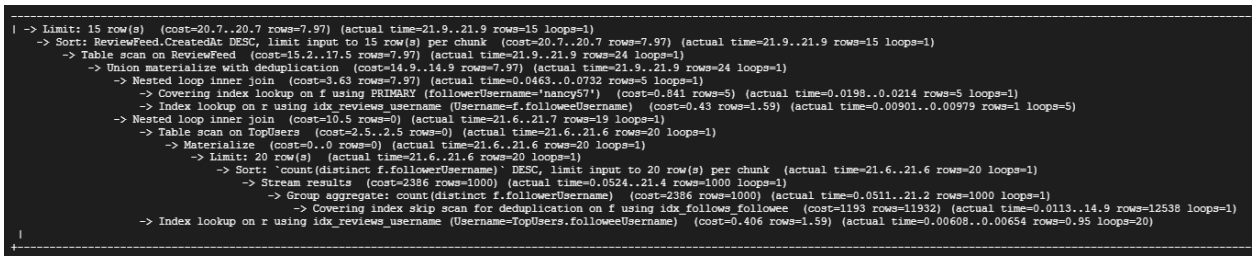
Design 2

CREATE INDEX rating ON Reviews(ReviewRating);
CREATE INDEX followerUsername ON Follower(FollowerUsername);



Cost = 15.2

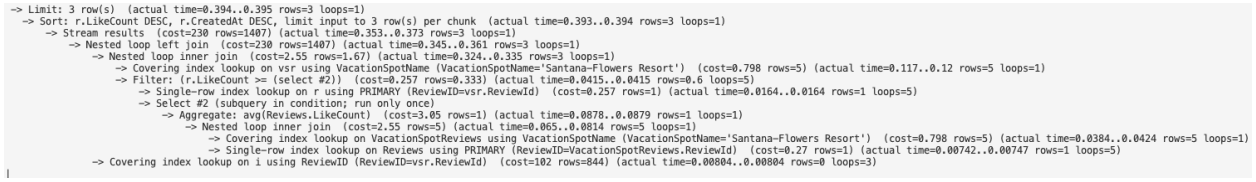
Design 3
CREATE INDEX rating ON Reviews(ReviewRating);
CREATE INDEX followerUsername ON Follower(FollowerUsername);
CREATE INDEX followeeUsername ON FavoriteSpots(Username);



Cost = 15.2

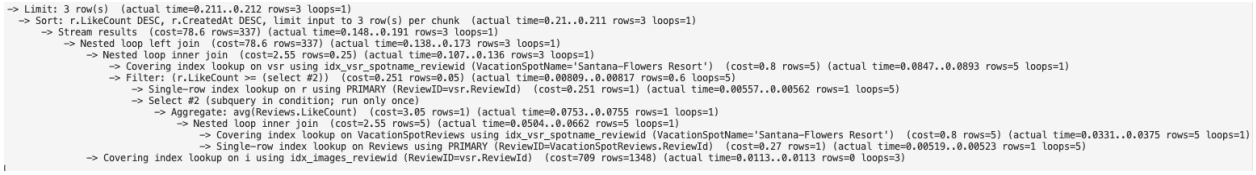
Design 3 was the best because it optimizes the two most important aspects of the query: filtering by followerUsername in Follows and joining Reviews by Username. The indexes reduce the number of rows scanned, significantly improving the efficiency of both the JOIN operations and the filtering process, leading to the lowest query cost.

-- Query 2
Without indexing:



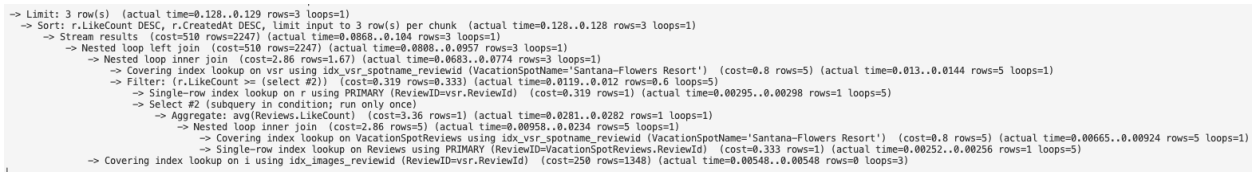
Cost is 230

Design 1:
CREATE INDEX idx_vsr_spotname_reviewid ON VacationSpotReviews(VacationSpotName, ReviewID);



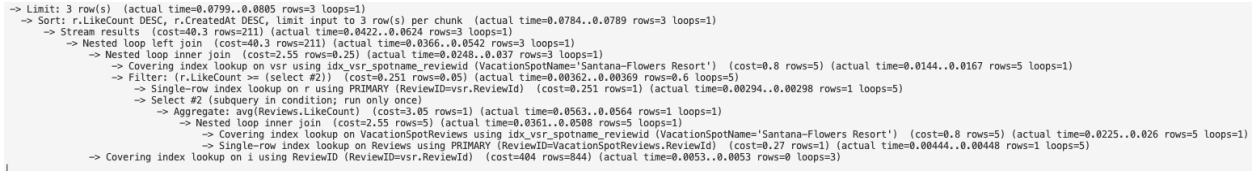
Cost is 78.6

Design 2:
CREATE INDEX idx_reviews_likecount_createdat ON Reviews (LikeCount, CreatedAt);



Cost is 510

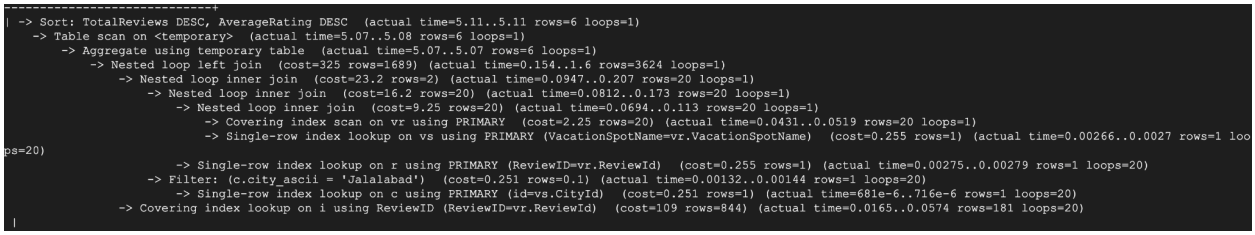
Design 3:
CREATE INDEX idx_vsr_spotname_reviewid ON VacationSpotReviews(VacationSpotName, ReviewID);
CREATE INDEX idx_images_reviewid ON Images (ReviewID);



Cost is 40.3

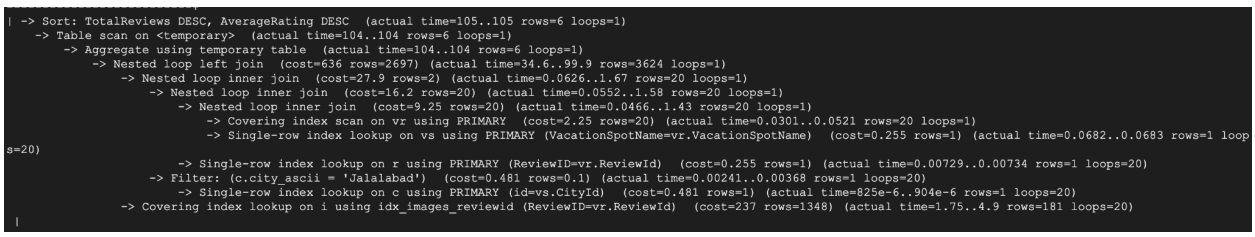
Design 3 was the best because it optimizes the two most important parts of the query: filtering by VacationSpotName in VacationSpotReviews and joining Images by ReviewID. The indexes reduce the number of rows the database has to scan, improving the performance of both the join operations and the filtering process, which results in the lowest cost.

-- Query 3
Without indexing



Cost is 325

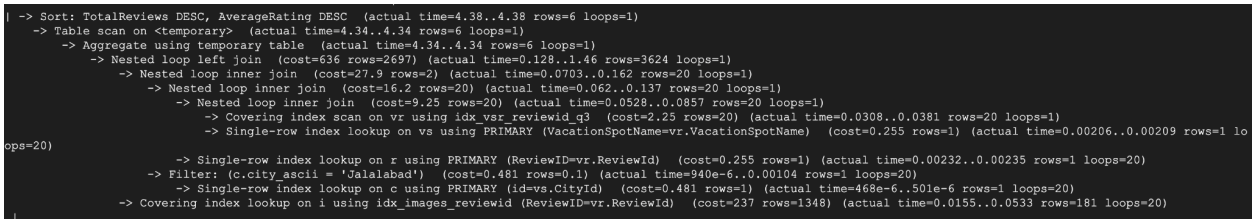
Design 1
CREATE INDEX CityId ON VacationSpots(CityId);



Cost is 636

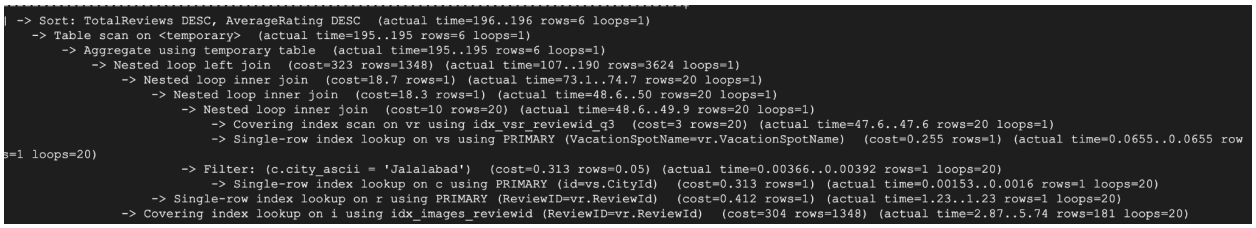
Design 2

CREATE INDEX idx_vsr_reviewid_q3 ON VacationSpotReviews(ReviewID);



Cost is 636

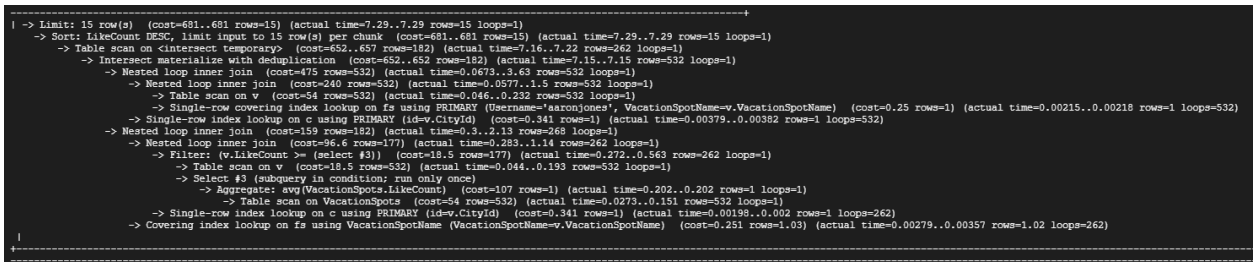
Design 3
CREATE INDEX asciiindex ON WorldCities(city_ascii)



Cost is 323

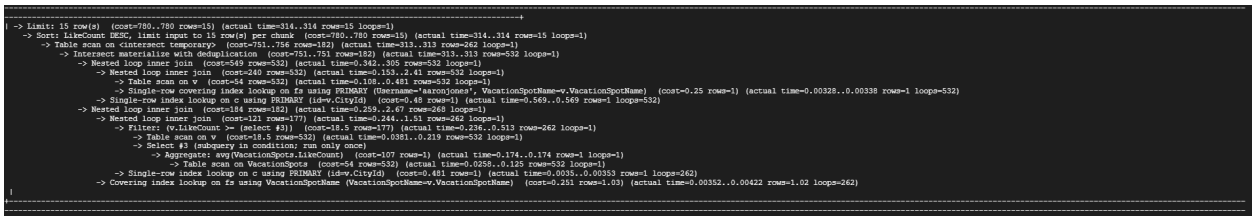
The design for this advanced query is design 3 since it decreases the cost by 2.It optimized the query so that when we were searching through worldcities it found faster searches.

-- Query 4
Before Indexing



Cost = 681

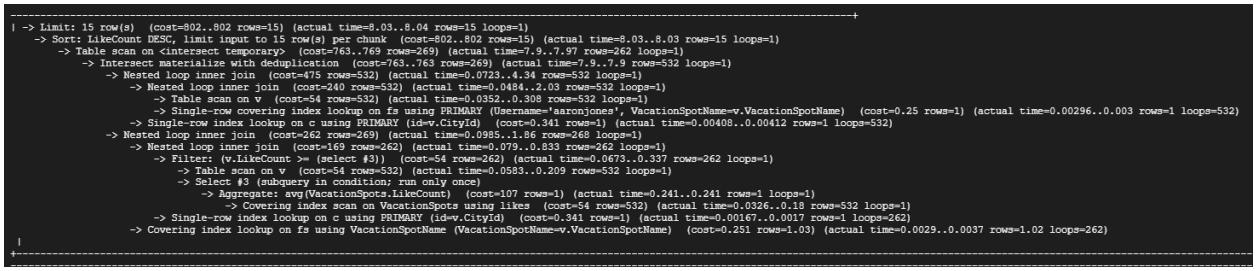
Design 1
CREATE INDEX idx_fs_username ON FavoriteSpots(Username);



Cost = 780

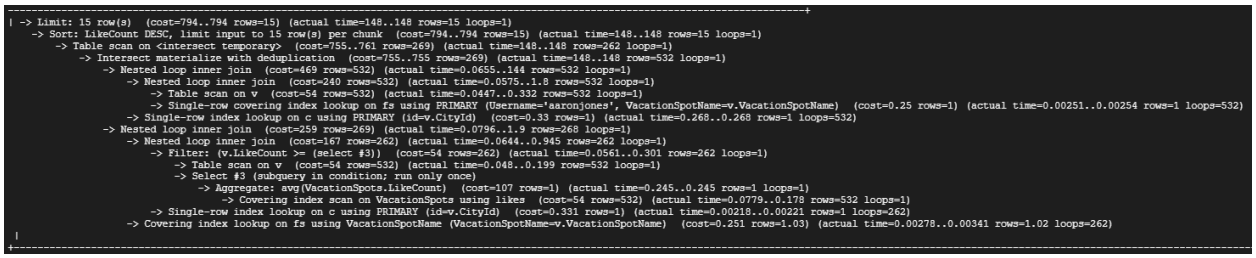
Design 2

CREATE INDEX idx_vs_likecount ON VacationSpots(LikeCount);



Cost = 802

Design 3
CREATE INDEX idx_vs_likecount ON VacationSpots(LikeCount);
CREATE INDEX idx_cities_cityid ON WorldCities(city_ascii);



Cost = 794

The design for this advanced query is the default indexing, without adding create index. When we added indexes, it actually increased the cost so we feel the best design in this case would just be going with the default indexing. Likely since the default indexing was picked as the create index, we actually created duplicate indexes here which took up extra memory/writing space rather than helping.