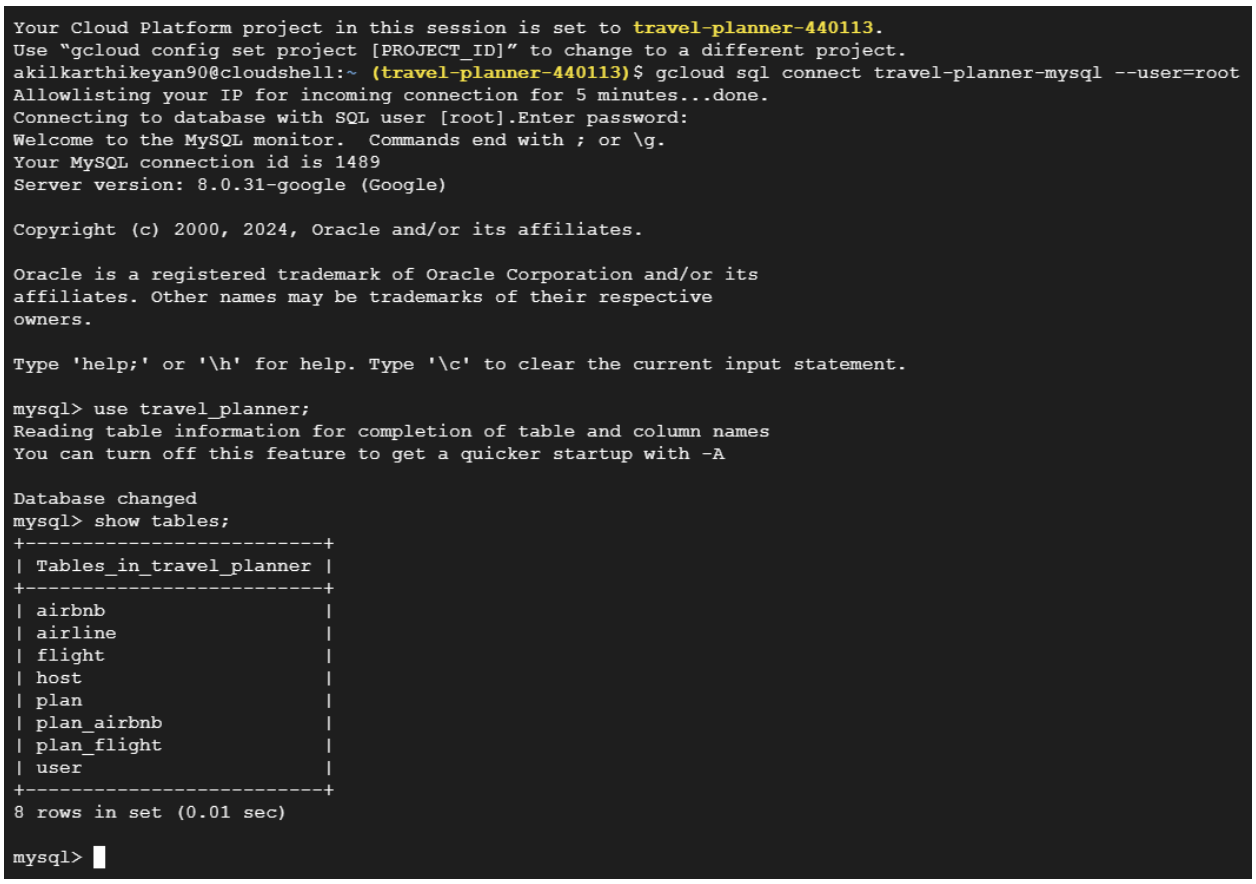
Database Implementation and Indexing

Screenshot of connection to GCP:



Data Definition Language (DDL):

```
DROP TABLE IF EXISTS plan_flight;
DROP TABLE IF EXISTS plan_airbnb;
DROP TABLE IF EXISTS plan;
DROP TABLE IF EXISTS airbnb;
DROP TABLE IF EXISTS flight;
DROP TABLE IF EXISTS user;
DROP TABLE IF EXISTS airline;
DROP TABLE IF EXISTS host;

CREATE TABLE host
(
    host_id             INT,
```

```
    host_url              VARCHAR(300),
    host_name             VARCHAR(50),
    host_response_rate    NUMERIC,
    host_acceptance_rate  NUMERIC,
    host_is_superhost     BOOL,
    host_identity_verified BOOL,
    PRIMARY KEY (host_id)
);

CREATE TABLE airline
(
    airline_id   VARCHAR(2),
    airline_name VARCHAR(50),
    PRIMARY KEY (airline_id)
);

CREATE TABLE user
(
    user_id   INT AUTO_INCREMENT,
    user_name VARCHAR(50) NOT NULL,
    phone     NUMERIC,
    email     VARCHAR(50),
    PRIMARY KEY (user_id)
);

CREATE TABLE flight
(
    flight_id            VARCHAR(50),
    flight_date          DATE          NOT NULL,
    starting_airport     VARCHAR(3)    NOT NULL,
    destination_airport  VARCHAR(3)    NOT NULL,
    travel_duration      NUMERIC,
    is_basic_economy     BOOL,
    total_fare           DECIMAL(10, 2) NOT NULL,
    departure_time       TIME          NOT NULL,
    arrival_time         TIME          NOT NULL,
    equipment_description VARCHAR(50),
    airline_id           VARCHAR(2)    NOT NULL,
    FOREIGN KEY (airline_id) REFERENCES airline (airline_id),
    PRIMARY KEY (flight_id)
);

CREATE TABLE airbnb
(
```

```
    airbnb_id             INT,
    listing_url           VARCHAR(300),
    name                  VARCHAR(150),
    description           VARCHAR(300),
    neighborhood_overview VARCHAR(300),
    picture_url           VARCHAR(300),
    latitude              DECIMAL(8, 6),
    longitude             DECIMAL(9, 6),
    property_type         VARCHAR(150),
    accommodates          NUMERIC,
    bathrooms             NUMERIC,
    bedrooms              NUMERIC,
    beds                  NUMERIC,
    amenities             JSON,
    price                 NUMERIC     NOT NULL,
    number_of_reviews     NUMERIC,
    review_scores_rating  NUMERIC     NOT NULL,
    close_to_airport      VARCHAR(3)  NOT NULL,
    host_id               INT NOT NULL,
    FOREIGN KEY (host_id) REFERENCES host (host_id),
    PRIMARY KEY (airbnb_id)
);

CREATE TABLE plan
(
    plan_id          INT AUTO_INCREMENT,
    plan_name        VARCHAR(50) NOT NULL,
    plan_description VARCHAR(150),
    user_id          INT,
    FOREIGN KEY (user_id) REFERENCES user(user_id) ON DELETE CASCADE,
    PRIMARY KEY (plan_id)
);


CREATE TABLE plan_airbnb
(
    plan_id    INT,
    airbnb_id  INT,
    start_date DATE,
    end_date   DATE,
    ordinal    INT,
    FOREIGN KEY (plan_id) REFERENCES plan (plan_id) ON DELETE CASCADE,
    FOREIGN KEY (airbnb_id) REFERENCES airbnb (airbnb_id),
    PRIMARY KEY (plan_id, airbnb_id, ordinal)
```

```
);

CREATE TABLE plan_flight
(
   plan_id  INT,
   flight_id VARCHAR(50),
   ordinal   INT,
   FOREIGN KEY (plan_id) REFERENCES plan (plan_id) ON DELETE CASCADE,
   FOREIGN KEY (flight_id) REFERENCES flight (flight_id),
   PRIMARY KEY (plan_id, flight_id)
);
```

Count query on 3 tables:

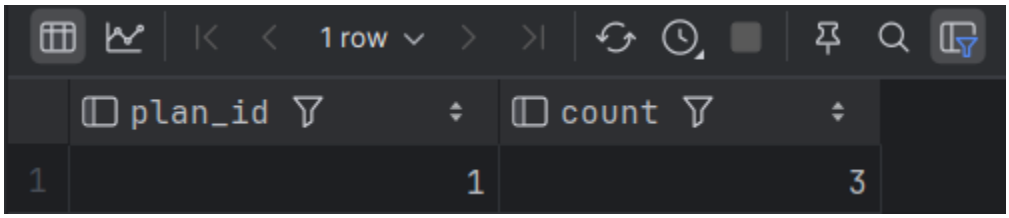
```
mysql> select count(*) from airbnb;
+----------+
| count(*) |
+----------+
|    16000 |
+----------+
1 row in set (7.76 sec)

mysql> select count(*) from host;
+----------+
| count(*) |
+----------+
|    10987 |
+----------+
1 row in set (0.02 sec)

mysql> select count(*) from flight;
+----------+
| count(*) |
+----------+
|    70221 |
+----------+
1 row in set (0.82 sec)

mysql>
```

Advanced Queries:

1. Query to find count of flights/airbnbs for every plan for a given user

```
select p.plan_id, (select count(*) from plan_airbnb pa where pa.plan_id = p.plan_id) +
                  (select count(*) from plan_flight pf where pf.plan_id = p.plan_id) count
from user u natural join plan p where
u.user_id = 1;
```

Output:



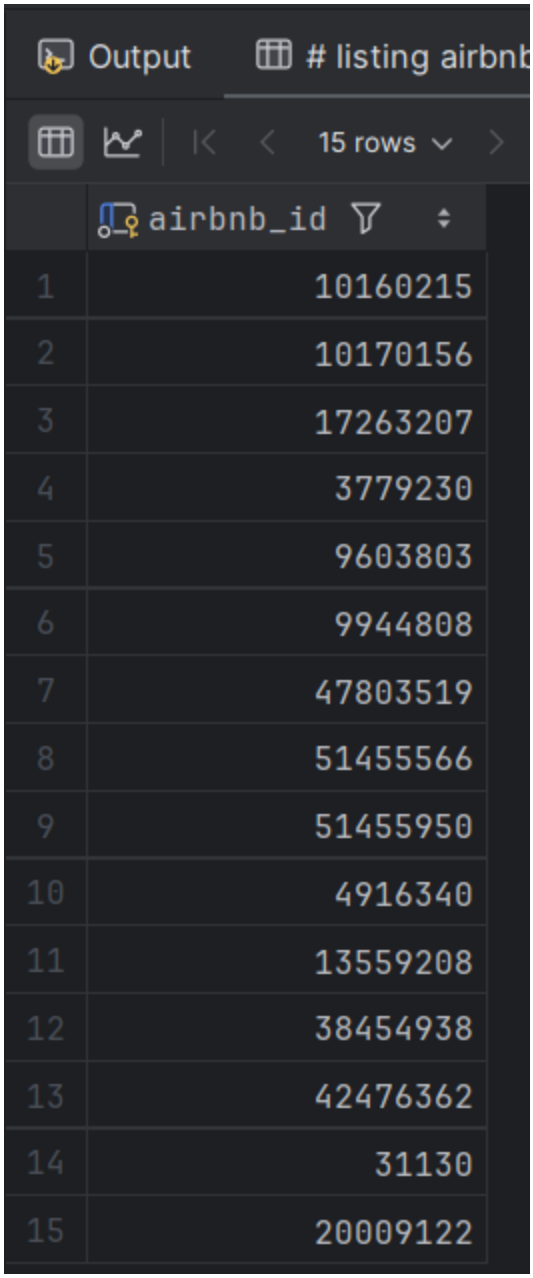
Explanation:

Each time a user creates a new plan, a unique plan_id is generated. Currently, only one plan has been inserted for user_id=1, which is why the query outputs only one record.

2. Query to find airbnbs close to a given airport, whose host owns at least 3 airbnbs

```
select a.airbnb_id
from airbnb a
        natural join host h
where a.close_to_airport = 'JFK'
  and a.host_id in
      (select h.host_id
       from host h
              natural join airbnb a
       group by h.host_id
       having count(*) >= 3)
       LIMIT 15;
```

Output:

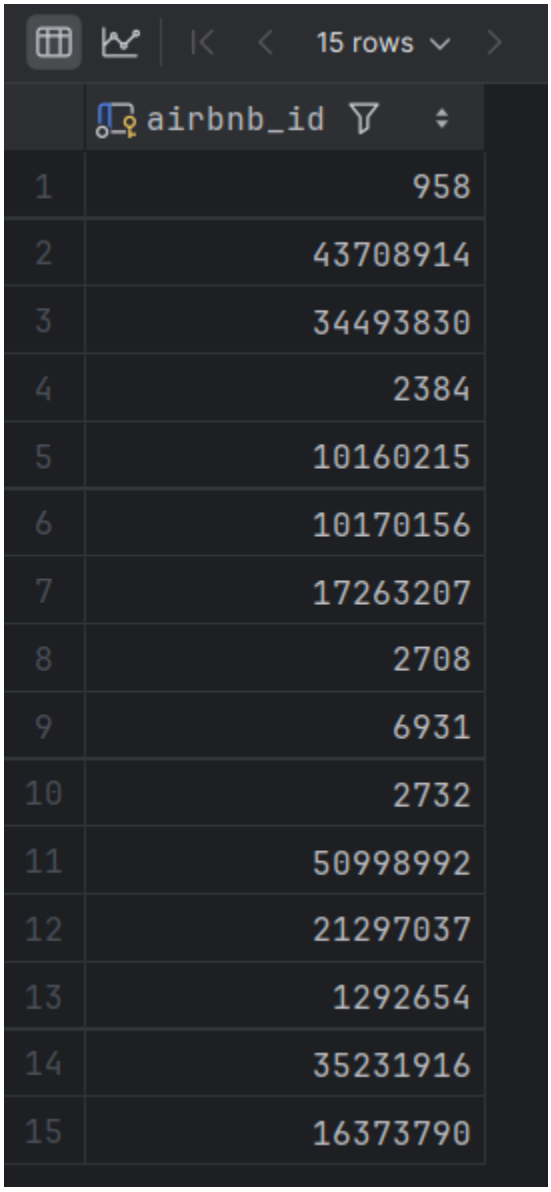


3. Query to find airbnbs close to a given airport, whose host response rate is greater than or equal to the average host response rate of all airbnbs close to the given airport

```
select a.airbnb_id
from airbnb a
        natural join host h
where a.close_to_airport = 'JFK'
  and h.host_response_rate >=
      (select AVG(h1.host_response_rate)
```

```
    FROM airbnb a1
        NATURAL JOIN host h1
    where a1.close_to_airport = 'JFK')
LIMIT 15;
```

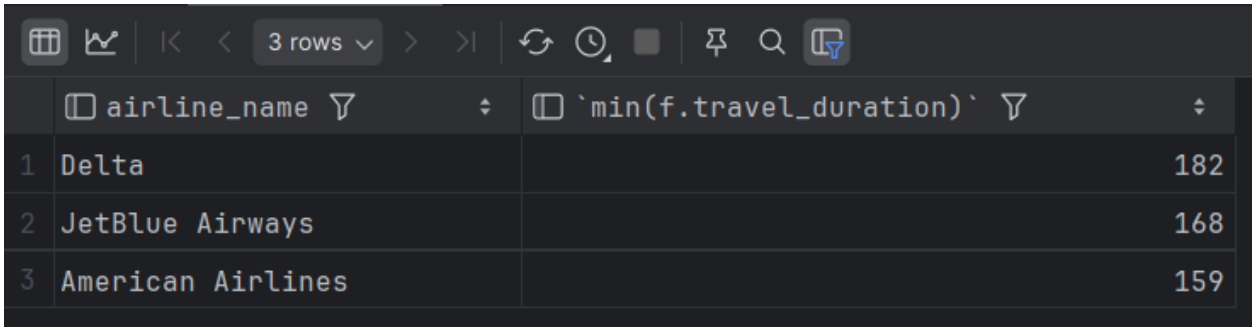
Output:



4. Query to find the quickest flight's duration for each airline from a specific starting airport to a specific destination airport on a specific date

```
select a.airline_name, min(f.travel_duration)
from flight f
        natural join airline a
where f.flight_date = '2022-07-03'
  and f.starting_airport = 'JFK'
  and f.destination_airport = 'ORD'
group by a.airline_id;
```

Output:



	airline_name	`min(f.travel_duration)`
1	Delta	182
2	JetBlue Airways	168
3	American Airlines	159

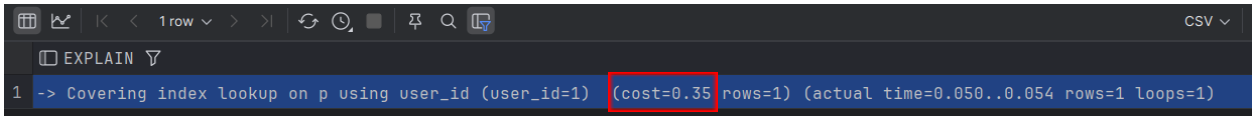
Explanation:

The query returns only 3 records because it applies specific filters that narrow down the data to flights on a particular route and date, and it groups the results by airline to show only the minimum travel duration per airline. Since only 3 airlines operate flights on July 3 2022, from JFK to ORD, the query returns only 3 records.

Indexing Advanced Queries:

1.

```
explain analyze select p.plan_id, (select count(*) from plan_airbnb pa where pa.plan_id =
p.plan_id) +
            (select count(*) from plan_flight pf where pf.plan_id = p.plan_id) count
from user u natural join plan p where
u.user_id = 1;
```

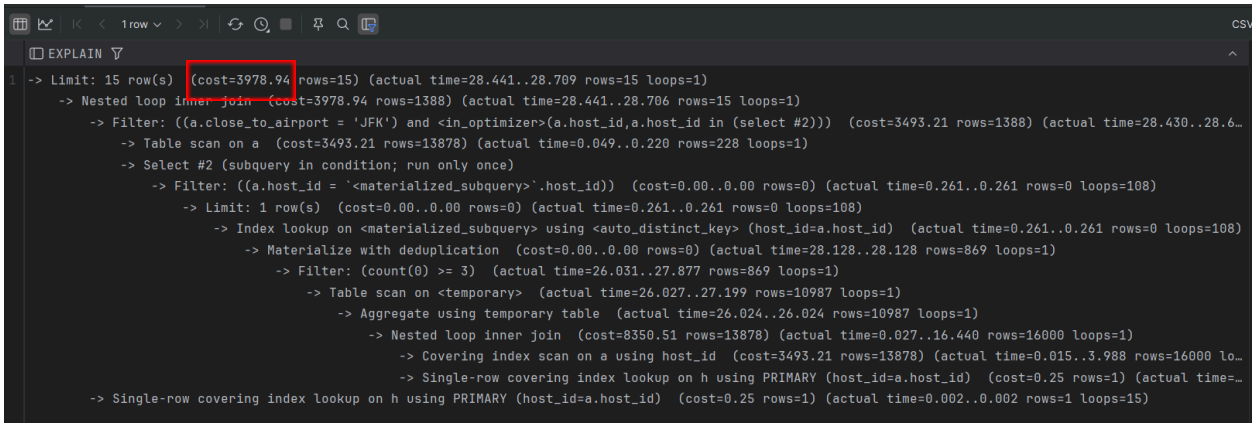


We did not do any indexing for this advanced query, because join attributes, where clause attributes and group by attributes are all primary keys that are already indexed.

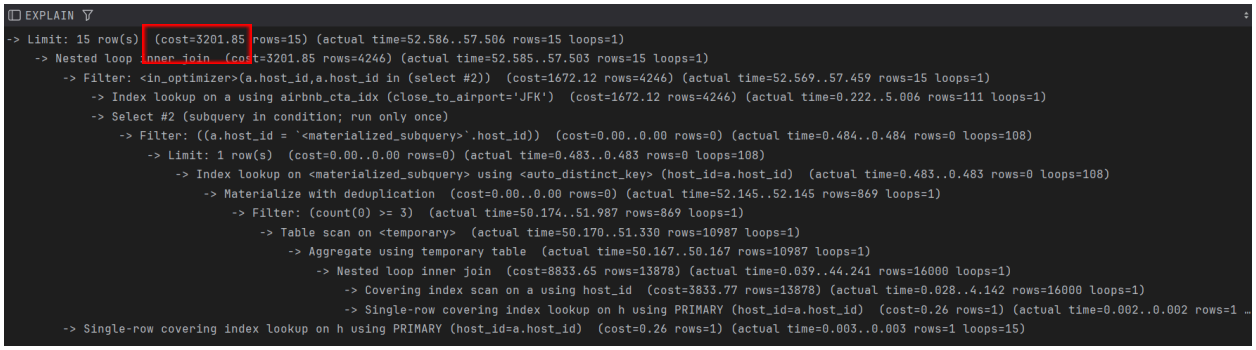
2.

```
select a.airbnb_id
from airbnb a
        natural join host h
where a.close_to_airport = 'JFK'
  and a.host_id in
      (select h.host_id
       from host h
              natural join airbnb a
       group by h.host_id
       having count(*) >= 3)
       LIMIT 15;
```

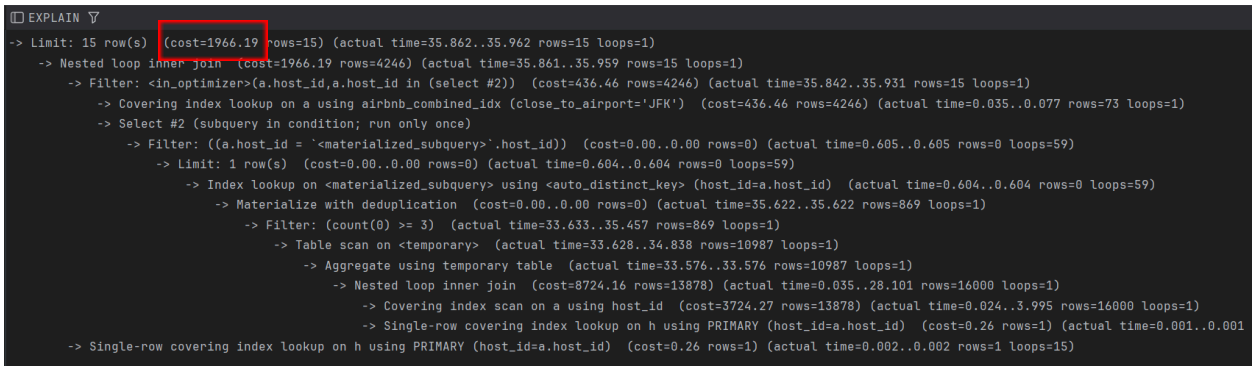
Before indexing:



After indexing only airbnb(close_to_airport):



After indexing on airbnb(close_to_airport, host_id):



Explanation:

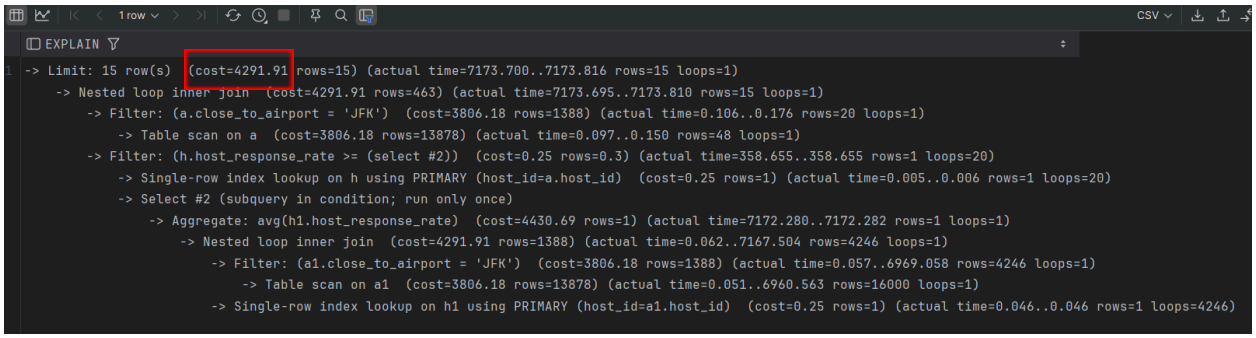
We examined the where clause attributes, join attributes and group by attributes for this query. The attributes being used are airbnb.close_to_airport and airbnb.host_id. We tried indexing close_to_airport and (close_to_airport, host_id). We omitted trying to index host_id alone, as this is already a foreign key and is indexed. Finally, we chose (close_to_airport, host_id) as the index, because this has lesser cost compared to the former as indicated by the boxes.

3.

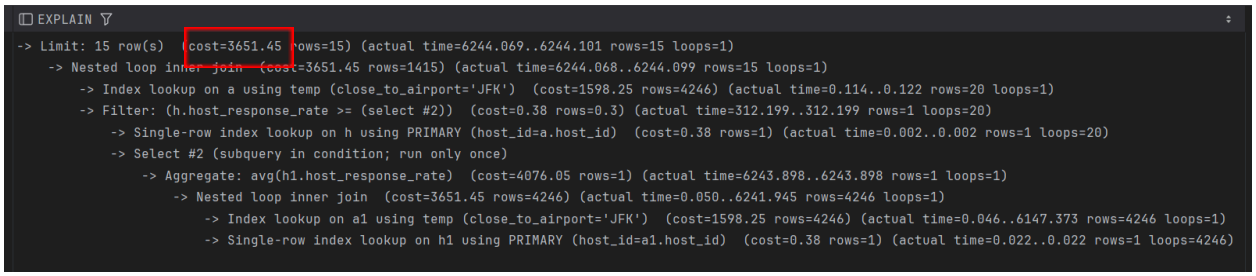
```
explain analyze
select a.airbnb_id
from airbnb a
        natural join host h
where a.close_to_airport = 'JFK'
  and h.host_response_rate >=
      (select AVG(h1.host_response_rate)
       FROM airbnb a1
```

```
        NATURAL JOIN host h1
    where a1.close_to_airport = 'JFK')
LIMIT 15;
```

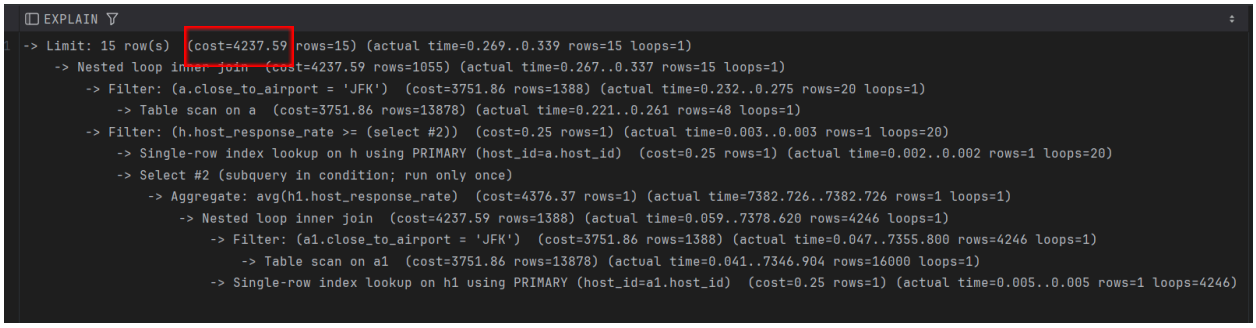
Before indexing:



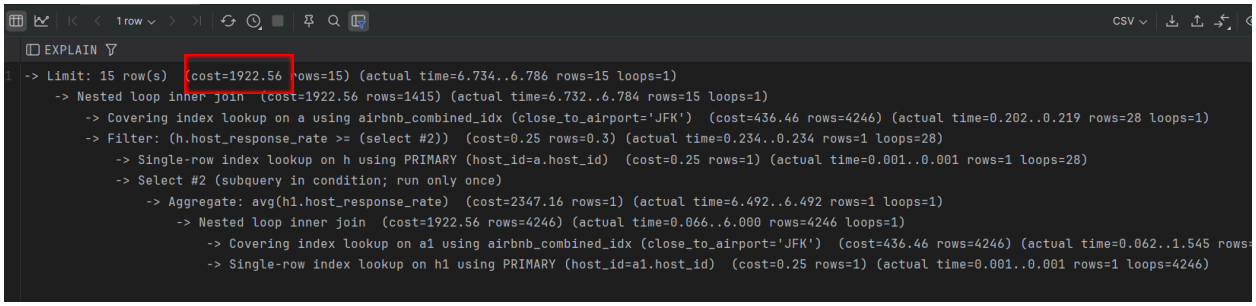
After indexing only airbnb(close_to_airport):



After indexing only host(host_response_rate):



After indexing airbnb(close_to_airport, host_id):



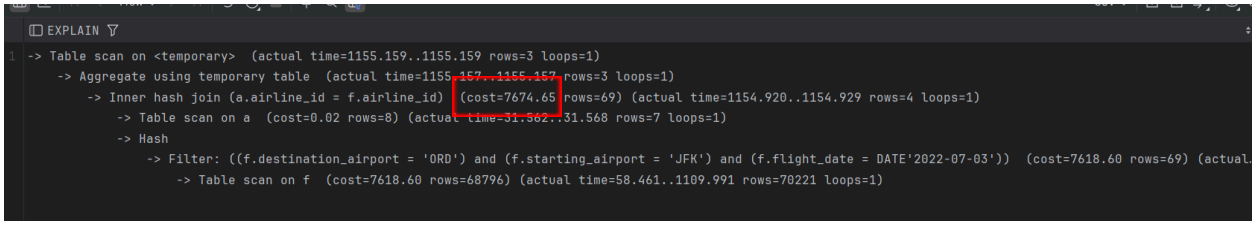
Explanation:

We examined the where clause attributes, join attributes and group by attributes for this query. The attributes being used are airbnb.close_to_airport, airbnb.host_id and host.host_response_rate. After trying to index various combinations of these attributes, we settled for airbnb(close_to_airport, host_id) because this results in the lowest cost (also this works best for the previous query too).

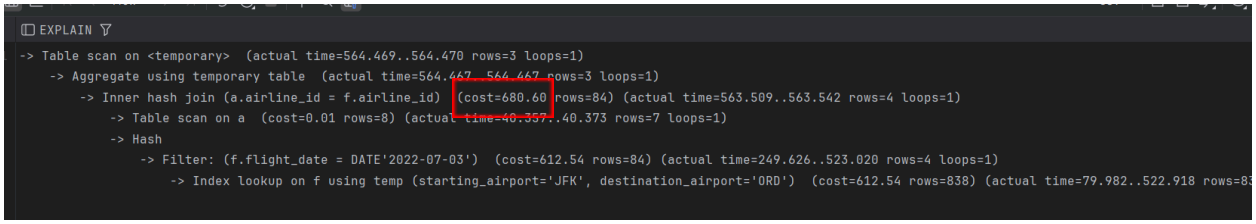
4.

```
explain analyze
select a.airline_name, min(f.travel_duration)
from flight f
        natural join airline a
where f.flight_date = '2022-07-03'
  and f.starting_airport = 'JFK'
  and f.destination_airport = 'ORD'
group by a.airline_id;
```

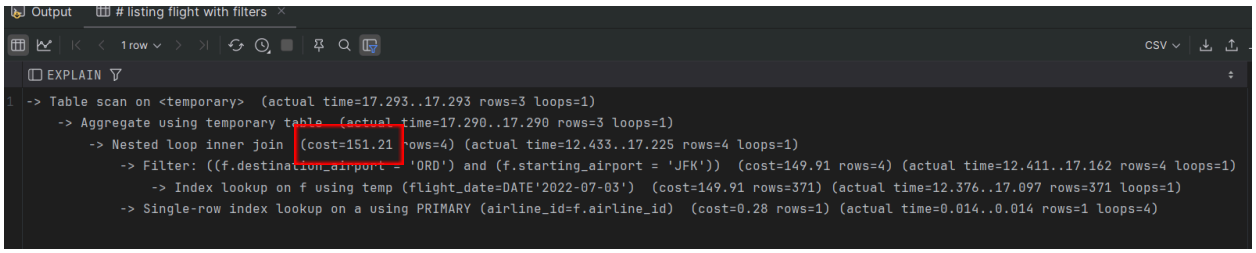
Before indexing:



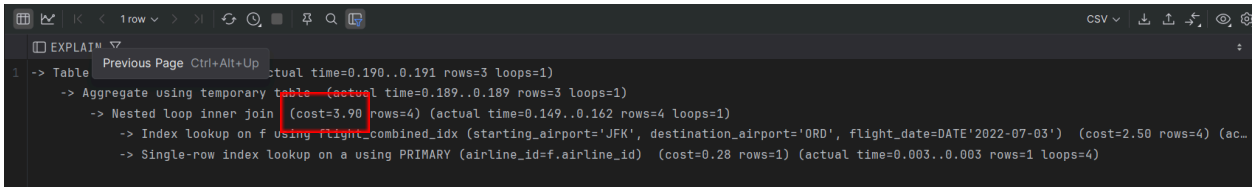
After indexing only flight(starting_airport, destination_airport):



After indexing only flight(flight_date):



After indexing flight(starting_airport, destination_airport, flight_date):



Explanation:

The attributes being used in the where clause are flight.starting_airport, flight.destination_airport and flight.flight_date. We tried indexing various combinations of these and indexing flight(starting_airport, destination_airport, flight_date) gave us the best result by far. Also, this works well with our use case because other flight filters that we plan to implement for our application will also contain flight.starting_airport, flight.destination_airport and flight.flight_date in the where clause.