Part 1

Database connection on GCP:

```
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to flightdelaypro.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
jackwangcas@cloudshell:~ (flightdelaypro)$ gcloud sql connect flightdatabase --user=root
Allowlisting your IP for incoming connection for 5 minutes...done.
Connecting to database with SQL user [root].Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 410
Server version: 8.0.31-google (Google)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Data Definition Language (DDL) commands:

```
CREATE TABLE User(
   user_id INT,
   username VARCHAR(255),
   password INT,

   PRIMARY KEY (user_id)
);

CREATE TABLE Delay(
   delay_number INT,
   user_id INT,
   flight_number INT,
   airline_code VARCHAR(2),
   minutes INT,
   day_of_week INT,
   distance INT,

   PRIMARY KEY (delay_number),
   FOREIGN KEY (user_id) REFERENCES User(user_id) ON DELETE CASCADE,
   FOREIGN KEY (flight_number) REFERENCES Operate_Flight(flight_number) ON
DELETE CASCADE,
   FOREIGN KEY (airline_code) REFERENCES Airline(airline_code) ON DELETE
CASCADE
);

CREATE TABLE Operate_Flight(
   flight_number INT,
   airline_code VARCHAR(2),
   orig_airport_code VARCHAR(4),
   dest_airport_code VARCHAR(4),

   PRIMARY KEY (flight_number, airline_code),
   FOREIGN KEY (airline_code) REFERENCES Airline(airline_code) ON DELETE
CASCADE,
   FOREIGN KEY (orig_airport_code) REFERENCES Airport(airport_code) ON
DELETE CASCADE,
   FOREIGN KEY (dest_airport_code) REFERENCES Airport(airport_code) ON
DELETE CASCADE
```

```sql
);

CREATE TABLE Airport(
    airport_code VARCHAR(4),
    airport_name VARCHAR(255),
    city VARCHAR(255),
    state VARCHAR(2),

    PRIMARY KEY (airport_code)
);

CREATE TABLE Airline(
    airline_code VARCHAR(2),
    airline_name VARCHAR(255),

    PRIMARY KEY (airline_code)
);
```

Inserting rows into tables:

```
Use flightdatabase;

SELECT COUNT(*) FROM Operate_Flight
```

100% 36:3

Result Grid Filter Rows: Search Export:

COUNT(*)
17416

Limit to 1000 rows

```
Use flightdatabase;

SELECT COUNT(*) FROM User;
```

100% 81:32

Result Grid Filter Rows: Search Export:

COUNT(*)
1077

```
Use flightdatabase;

SELECT COUNT(*) FROM Delay
```

100% 27:3

Result Grid Filter Rows: Search Export:

COUNT(*)
1048575

Advanced SQL Queries

Query 1:

The sql retrieves flights between start and destination cities(Seattle and San Jose in this case). It gives details about the flight, airlines operating them, and the number of delays and cancellations for those flights.

```sql
SELECT
    Operate_Flight.flight_number,
    Operate_Flight.airline_code,
    Operate_Flight.orig_airport_code,
    Operate_Flight.dest_airport_code,
    Airline.airline_name,
    COUNT(Delay.delay_number) AS num_delays,
    SUM(CASE WHEN Delay.minutes IS NOT NULL THEN 1 ELSE 0 END) AS num_cancellations
FROM Operate_Flight
INNER JOIN Airport AS start_airport ON Operate_Flight.orig_airport_code = start_airport.airport_code
INNER JOIN Airport AS destination_airport ON Operate_Flight.dest_airport_code = destination_airport.airport_code
INNER JOIN Airline ON Operate_Flight.airline_code = Airline.airline_code
LEFT JOIN Delay ON Operate_Flight.flight_number = Delay.flight_number
WHERE start_airport.city = 'Seattle' -- Replace with the actual city name for the start airport
    AND destination_airport.city = 'San Jose' -- Replace with the actual city name for the destination airport
GROUP BY
    Operate_Flight.flight_number,
    Operate_Flight.airline_code,
    Operate_Flight.orig_airport_code,
    Operate_Flight.dest_airport_code,
    Airline.airline_name
ORDER BY num_delays ASC, num_cancellations ASC;
```

Results:

```sql
Use flightdatabase;


SELECT
    Operate_Flight.flight_number,
    Operate_Flight.airline_code,
    Operate_Flight.orig_airport_code,
    Operate_Flight.dest_airport_code,
    Airline.airline_name,
    COUNT(Delay.delay_number) AS num_delays,
    SUM(CASE WHEN Delay.minutes IS NOT NULL THEN 1 ELSE 0 END) AS num_cancellations
FROM Operate_Flight
INNER JOIN Airport AS start_airport ON Operate_Flight.orig_airport_code = start_airport.airport_code
INNER JOIN Airport AS destination_airport ON Operate_Flight.dest_airport_code = destination_airport.airport_code
INNER JOIN Airline ON Operate_Flight.airline_code = Airline.airline_code
LEFT JOIN Delay ON Operate_Flight.flight_number = Delay.flight_number
WHERE start_airport.city = 'Seattle' -- Replace with the actual city name for the start airport
    AND destination_airport.city = 'San Jose' -- Replace with the actual city name for the destination airport
GROUP BY
    Operate_Flight.flight_number,
    Operate_Flight.airline_code,
    Operate_Flight.orig_airport_code,
    Operate_Flight.dest_airport_code,
    Airline.airline_name
ORDER BY num_delays ASC, num_cancellations ASC;
```

Result Grid

flight_numb...	airline_code	orig_airport_co...	dest_airport_co...	airline_name	num_delays	num_cancellatio...
3858	WN	SEA	SJC	Southwest Airlines Co.	19	19
4802	OO	SEA	SJC	Skywest Airlines Inc.	66	66
4733	OO	SEA	SJC	Skywest Airlines Inc.	161	161
4571	WN	SEA	SJC	Southwest Airlines Co.	201	201
4727	OO	SEA	SJC	Skywest Airlines Inc.	240	240
895	WN	SEA	SJC	Southwest Airlines Co.	252	252
200	AS	SEA	SJC	Alaska Airlines Inc.	408	408
324	AS	SEA	SJC	Alaska Airlines Inc.	464	464
330	AS	SEA	SJC	Alaska Airlines Inc.	470	470
336	AS	SEA	SJC	Alaska Airlines Inc.	476	476
322	AS	SEA	SJC	Alaska Airlines Inc.	484	484
326	AS	SEA	SJC	Alaska Airlines Inc.	486	486
334	AS	SEA	SJC	Alaska Airlines Inc.	570	570
338	AS	SEA	SJC	Alaska Airlines Inc.	573	573
328	AS	SEA	SJC	Alaska Airlines Inc.	607	607

Query 2:

Retrieves information about airlines, including the name of the airline, the number of delays, and the number of cancellations.

```
SELECT
    Airline.airline_name,
    SUM(CASE WHEN Delay.minutes IS NOT NULL THEN 1 ELSE 0 END) AS num_delays,
    COUNT(Delay.delay_number) AS num_cancellations
FROM Airline
LEFT JOIN Operate_Flight ON Airline.airline_code = Operate_Flight.airline_code
LEFT JOIN Delay ON Operate_Flight.flight_number = Delay.flight_number
GROUP BY Airline.airline_name;
```

Result:

Limit to 1000 rows

```
Use flightdatabase;

SELECT
    Airline.airline_name,
    SUM(CASE WHEN Delay.minutes IS NOT NULL THEN 1 ELSE 0 END) AS num_delays,
    COUNT(Delay.delay_number) AS num_cancellations
FROM Airline
LEFT JOIN Operate_Flight ON Airline.airline_code = Operate_Flight.airline_code
LEFT JOIN Delay ON Operate_Flight.flight_number = Delay.flight_number
GROUP BY Airline.airline_name LIMIT 0, 15;
```

100% 20:1

Result Grid Filter Rows: Search Export:

airline_name	num_delays	num_cancellatio...
Alaska Airlines Inc.	207199	207199
American Airlines Inc.	347673	347673
American Eagle Airlines Inc.	93823	93823
Atlantic Southeast Airlines	209352	209352
Delta Air Lines Inc.	565927	565927
Frontier Airlines Inc.	170670	170670
Hawaiian Airlines Inc.	99195	99195
JetBlue Airways	288323	288323
Skywest Airlines Inc.	183029	183029
Southwest Airlines Co.	876407	876407
Spirit Air Lines	111996	111996
United Air Lines Inc.	413710	413710
US Airways Inc.	254144	254144
Virgin America	69324	69324

Part 2

Query 1

Query 1 before any indexing:

```sql
EXPLAIN ANALYZE
SELECT
    Operate_Flight.flight_number,
    Operate_Flight.airline_code,
    Operate_Flight.orig_airport_code,
    Operate_Flight.dest_airport_code,
    Airline.airline_name,
    COUNT(Delay.delay_number) AS num_delays,
    SUM(CASE WHEN Delay.minutes IS NOT NULL THEN 1 ELSE 0 END) AS num_cancellations
FROM Operate_Flight
INNER JOIN Airport AS start_airport ON Operate_Flight.orig_airport_code = start_airport.airport_code
INNER JOIN Airport AS destination_airport ON Operate_Flight.dest_airport_code = destination_airport.airport_code
INNER JOIN Airline ON Operate_Flight.airline_code = Airline.airline_code
LEFT JOIN Delay ON Operate_Flight.flight_number = Delay.flight_number
WHERE start_airport.city = 'Seattle' -- Replace with the actual city name for the start airport
    AND destination_airport.city = 'San Jose' -- Replace with the actual city name for the destination airport
GROUP BY
    Operate_Flight.flight_number,
    Operate_Flight.airline_code,
    Operate_Flight.orig_airport_code,
    Operate_Flight.dest_airport_code,
    Airline.airline_name
ORDER BY num_delays ASC, num_cancellations ASC;
```

100% 48:23

Form Editor Navigate: 1/1

EXPLAIN:

```
-> Nested loop left join  (cost=12173.43 rows=29535) (actual time=0.785..15.028 rows=5477 loops=1)
    -> Nested loop inner join  (cost=1836.32 rows=172) (actual time=0.525..1.307 rows=15 loops=1)
        -> Nested loop inner join  (cost=1234.36 rows=1720) (actual time=0.494..0.954 rows=127 loops=1)
            -> Nested loop inner join  (cost=632.41 rows=1720) (actual time=0.485..0.753 rows=127 loops=1)
```

Form Editor

Result 38 Read Only

Indexing design 1: add Airport.city as index

```sql
EXPLAIN ANALYZE
SELECT
    Operate_Flight.flight_number,
    Operate_Flight.airline_code,
    Operate_Flight.orig_airport_code,
    Operate_Flight.dest_airport_code,
    Airline.airline_name,
    COUNT(Delay.delay_number) AS num_delays,
    SUM(CASE WHEN Delay.minutes IS NOT NULL THEN 1 ELSE 0 END) AS num_cancellations
FROM Operate_Flight
INNER JOIN Airport AS start_airport ON Operate_Flight.orig_airport_code = start_airport.airport_code
INNER JOIN Airport AS destination_airport ON Operate_Flight.dest_airport_code = destination_airport.airport_code
INNER JOIN Airline ON Operate_Flight.airline_code = Airline.airline_code
LEFT JOIN Delay ON Operate_Flight.flight_number = Delay.flight_number
WHERE start_airport.city = 'Seattle' -- Replace with the actual city name for the start airport
    AND destination_airport.city = 'San Jose' -- Replace with the actual city name for the destination airport
GROUP BY
    Operate_Flight.flight_number,
    Operate_Flight.airline_code,
    Operate_Flight.orig_airport_code,
    Operate_Flight.dest_airport_code,
    Airline.airline_name
ORDER BY num_delays ASC, num_cancellations ASC;
```

100% 48:23

Form Editor Navigate: 1/1

EXPLAIN:

```
-> Nested loop left join  (cost=32.79 rows=34) (actual time=0.456..12.196 rows=5477 loops=1)
    -> Nested loop inner join  (cost=21.04 rows=0.2) (actual time=0.235..0.581 rows=15 loops=1)
        -> Nested loop inner join  (cost=20.97 rows=0.2) (actual time=0.228..0.510 rows=15 loops=1)
            -> Nested loop inner join  (cost=0.70 rows=1) (actual time=0.015..0.021 rows=1 loops=1)
```

Result Grid

Result 39 Read Only

Indexing design 2: add Airline.airline_name as index

```sql
EXPLAIN ANALYZE
SELECT
    Operate_Flight.flight_number,
    Operate_Flight.airline_code,
    Operate_Flight.orig_airport_code,
    Operate_Flight.dest_airport_code,
    Airline.airline_name,
    COUNT(Delay.delay_number) AS num_delays,
    SUM(CASE WHEN Delay.minutes IS NOT NULL THEN 1 ELSE 0 END) AS num_cancellations
FROM Operate_Flight
INNER JOIN Airport AS start_airport ON Operate_Flight.orig_airport_code = start_airport.airport_code
INNER JOIN Airport AS destination_airport ON Operate_Flight.dest_airport_code = destination_airport.airport_code
INNER JOIN Airline ON Operate_Flight.airline_code = Airline.airline_code
LEFT JOIN Delay ON Operate_Flight.flight_number = Delay.flight_number
WHERE start_airport.city = 'Seattle' -- Replace with the actual city name for the start airport
    AND destination_airport.city = 'San Jose' -- Replace with the actual city name for the destination airport
GROUP BY
    Operate_Flight.flight_number,
    Operate_Flight.airline_code,
    Operate_Flight.orig_airport_code,
    Operate_Flight.dest_airport_code,
    Airline.airline_name
ORDER BY num_delays ASC, num_cancellations ASC;
```

100% 48:23

Form Editor Navigate: 1 / 1

EXPLAIN:

```
-> Nested loop left join  (cost=12173.43 rows=29535) (actual time=0.984..14.402 rows=5477 loops=1)
    -> Nested loop inner join  (cost=1836.32 rows=172) (actual time=0.595..1.493 rows=15 loops=1)
        -> Nested loop inner join  (cost=1234.36 rows=1720) (actual time=0.541..1.075 rows=127 loops=1)
            -> Nested loop inner join  (cost=632.41 rows=1720) (actual time=0.524..0.849 rows=127 loops=1)
```

Result Grid

Result 46 Read Only

Indexing design 3: add Delay.minutes as index

```sql
EXPLAIN ANALYZE
SELECT
    Operate_Flight.flight_number,
    Operate_Flight.airline_code,
    Operate_Flight.orig_airport_code,
    Operate_Flight.dest_airport_code,
    Airline.airline_name,
    COUNT(Delay.delay_number) AS num_delays,
    SUM(CASE WHEN Delay.minutes IS NOT NULL THEN 1 ELSE 0 END) AS num_cancellations
FROM Operate_Flight
INNER JOIN Airport AS start_airport ON Operate_Flight.orig_airport_code = start_airport.airport_code
INNER JOIN Airport AS destination_airport ON Operate_Flight.dest_airport_code = destination_airport.airport_code
INNER JOIN Airline ON Operate_Flight.airline_code = Airline.airline_code
LEFT JOIN Delay ON Operate_Flight.flight_number = Delay.flight_number
WHERE start_airport.city = 'Seattle' -- Replace with the actual city name for the start airport
    AND destination_airport.city = 'San Jose' -- Replace with the actual city name for the destination airport
GROUP BY
    Operate_Flight.flight_number,
    Operate_Flight.airline_code,
    Operate_Flight.orig_airport_code,
    Operate_Flight.dest_airport_code,
    Airline.airline_name
ORDER BY num_delays ASC, num_cancellations ASC;
```

100% 21:7

Form Editor Navigate: 1 / 1

EXPLAIN:

```
-> Nested loop left join  (cost=12173.43 rows=29535) (actual time=0.591..11.460 rows=5477 loops=1)
    -> Nested loop inner join  (cost=1836.32 rows=172) (actual time=0.366..0.979 rows=15 loops=1)
        -> Nested loop inner join  (cost=1234.36 rows=1720) (actual time=0.340..0.699 rows=127 loops=1)
            -> Nested loop inner join  (cost=632.41 rows=1720) (actual time=0.334..0.552 rows=127 loops=1)
```

Form Editor

Result 48 Read Only

After analyzing our query, it became apparent that optimizing indexing could significantly enhance query performance. We explored three distinct indexing strategies focusing on

Airport.city, Delay.minutes, and Airline.airline_name. During this exploration, it was evident that indexing by Airport.city had the most substantial impact on query cost reduction. We have a few speculations: 1. Filtering on Airport.city optimizes WHERE clause conditions. 2. It improves join performance in multiple joins with the Airport table.

Query 2

Query 2 before indexing:

```
'-> Table scan on <temporary>  (actual time=24325.975..24325.980 rows=14 loops=1)
    -> Aggregate using temporary table  (actual time=24325.968..24325.968 rows=14 loops=1)
        -> Nested loop left join  (cost=1063385.13 rows=3033015) (actual time=0.351..8375.789 rows=3890772 loops=1)
            -> Nested loop left join  (cost=1829.94 rows=17662) (actual time=0.079..16.745 rows=17416 loops=1)
                -> Table scan on Airline  (cost=1.65 rows=14) (actual time=0.030..0.076 rows=14 loops=1)
                -> Covering index lookup on Operate_Flight using idx_airline_code (airline_code=Airline.airline_code)  (cost=13.45
rows=1262) (actual time=0.039..1.049 rows=1244 loops=14)
            -> Index lookup on Delay using idx_flight_number (flight_number=Operate_Flight.flight_number)  (cost=42.93 rows=172)
(actual time=0.170..0.455 rows=223 loops=17416)
'
```

Indexing design 1: add Airline.airline_name as index

```
'-> Group aggregate: sum((case when (Delay.minutes is not null) then 1 else 0 end)), count(Delay.delay_number)  (cost=1366686.61
rows=3033015) (actual time=577.601..10477.780 rows=14 loops=1)
    -> Nested loop left join  (cost=1063385.13 rows=3033015) (actual time=0.356..8193.782 rows=3890772 loops=1)
        -> Nested loop left join  (cost=1829.94 rows=17662) (actual time=0.072..15.547 rows=17416 loops=1)
            -> Covering index scan on Airline using airline_name_idx  (cost=1.65 rows=14) (actual time=0.017..0.065 rows=14 loops=1)
            -> Covering index lookup on Operate_Flight using idx_airline_code (airline_code=Airline.airline_code)  (cost=13.45
rows=1262) (actual time=0.044..0.971 rows=1244 loops=14)
        -> Index lookup on Delay using idx_flight_number (flight_number=Operate_Flight.flight_number)  (cost=42.93 rows=172)
(actual time=0.171..0.446 rows=223 loops=17416)
'
```

Indexing design 2: add Delay.minutes as index

```
'-> Table scan on <temporary>  (actual time=24775.125..24775.131 rows=14 loops=1)
    -> Aggregate using temporary table  (actual time=24775.119..24775.119 rows=14 loops=1)
        -> Nested loop left join  (cost=1063385.13 rows=3033015) (actual time=0.353..8332.079 rows=3890772 loops=1)
            -> Nested loop left join  (cost=1829.94 rows=17662) (actual time=0.072..17.240 rows=17416 loops=1)
                -> Table scan on Airline  (cost=1.65 rows=14) (actual time=0.023..0.070 rows=14 loops=1)
                -> Covering index lookup on Operate_Flight using idx_airline_code (airline_code=Airline.airline_code)  (cost=13.45
rows=1262) (actual time=0.038..1.095 rows=1244 loops=14)
            -> Index lookup on Delay using idx_flight_number (flight_number=Operate_Flight.flight_number)  (cost=42.93 rows=172)
(actual time=0.170..0.452 rows=223 loops=17416)
'
```

Indexing design 3: add Delay.day_of_week as index

```
'-> Table scan on <temporary>  (actual time=24849.646..24849.652 rows=14 loops=1)
    -> Aggregate using temporary table  (actual time=24849.640..24849.640 rows=14 loops=1)
        -> Nested loop left join  (cost=1063385.13 rows=3033015) (actual time=0.410..8480.640 rows=3890772 loops=1)
            -> Nested loop left join  (cost=1829.94 rows=17662) (actual time=0.104..18.295 rows=17416 loops=1)
                -> Table scan on Airline  (cost=1.65 rows=14) (actual time=0.047..0.092 rows=14 loops=1)
                -> Covering index lookup on Operate_Flight using idx_airline_code (airline_code=Airline.airline_code)  (cost=13.45
rows=1262) (actual time=0.043..1.153 rows=1244 loops=14)
            -> Index lookup on Delay using idx_flight_number (flight_number=Operate_Flight.flight_number)  (cost=42.93 rows=172)
(actual time=0.174..0.462 rows=223 loops=17416)
'
```

We don’t think any of these indexing design choices reduce the cost of our query. We think it’s because all of our joins used primary keys, so adding extra index is not necessary.