CS 411 Stage3

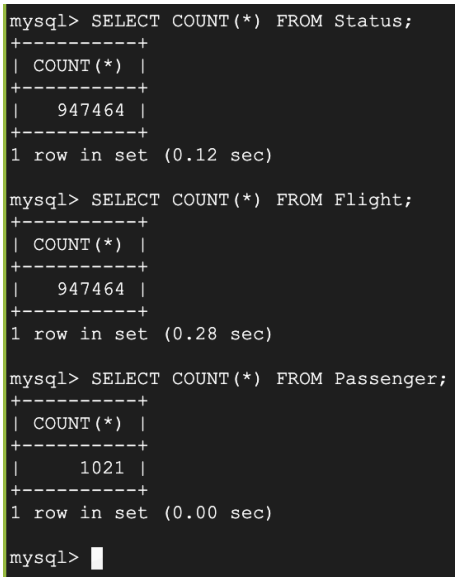
DDL Commands

```
CREATE TABLE Airport (
   IATACode VARCHAR(3) PRIMARY KEY,
   City VARCHAR(50) NOT NULL,
   State VARCHAR(50) NOT NULL
);

CREATE TABLE Flight (
   Date DATE,
   FlightNumber VARCHAR(10),
   AirlineName VARCHAR(20),
   DeptAirportID VARCHAR(3),
   ArrvAirportID VARCHAR(3),
   DeptTime INT,
   ArrvTime INT,
   ActualDeptTime INT,
   ActualArrvTime INT,
   PRIMARY KEY (Date, DeptTime, FlightNumber),
   FOREIGN KEY (DeptAirportID) REFERENCES Airport(IATACode),
   FOREIGN KEY (ArrvAirportID) REFERENCES Airport(IATACode)
);

CREATE TABLE Status (
   Date DATE,
   DeptTime INT,
   FlightNumber VARCHAR(10),
   FlightStatus VARCHAR(20) NOT NULL,
   PRIMARY KEY (Date, FlightNumber),
   FOREIGN KEY (Date, DeptTime, FlightNumber) REFERENCES Flight(Date, DeptTime,
FlightNumber)
);

CREATE TABLE Passenger (
   Email VARCHAR(100) PRIMARY KEY,
   FirstName VARCHAR(50) NOT NULL,
   LastName VARCHAR(50) NOT NULL
);


CREATE TABLE Booking (
```

```
    BookingId INT PRIMARY KEY,
    FlightDate DATE,
    DeptTime INT,
    FlightNumber VARCHAR(10),
    Email VARCHAR(100),
    BookingDate DATE NOT NULL,
    Status VARCHAR(20) NOT NULL,
    FOREIGN KEY (FlightDate, DeptTime, FlightNumber) REFERENCES Flight(Date, DeptTime,
FlightNumber),
    FOREIGN KEY (Email) REFERENCES Passenger(Email)
);
```

Count Query (Number of Rows)

```
Select Count(*) From Status;
Select Count(*) From Flight;
Select Count(*) From Passenger;
```



SQL Queries

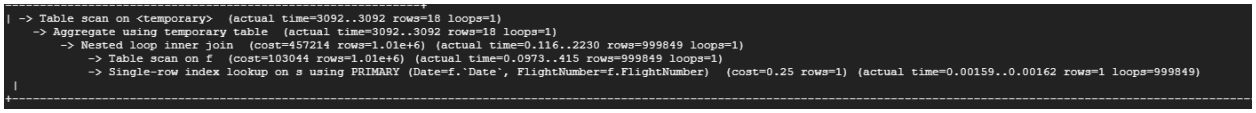
```
-- Airline Late Percentage

Select f.AirlineName, Count(Case When s.FlightStatus = 'DELAYED' Then 1 End) * 100 /
Count(f.FlightNumber) As late_percentage
```

From Flight f Join Status s on (s.Date = f.Date And s.FlightNumber = f.FlightNumber)
Group by f.AirlineName;

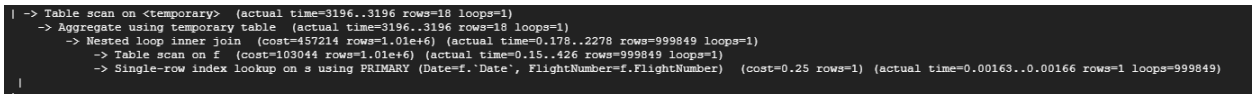
AirlineName	late_percentage
American Airlines In	34.7611
JetBlue Airways	39.6720
United Air Lines Inc	34.9765
Southwest Airlines C	45.3661
Delta Air Lines Inc.	30.6321
SkyWest Airlines Inc	24.2208
Spirit Air Lines	35.8585
PSA Airlines Inc.	26.7196
Envoy Air	26.2767
Alaska Airlines Inc.	32.1338
Frontier Airlines In	38.8016
Endeavor Air Inc.	19.4442
Mesa Airlines Inc.	27.0038
Republic Airline	20.4305
Hawaiian Airlines In	37.9310

15 rows in set (3.10 sec)

Explain analyze:
Cost: 0.25

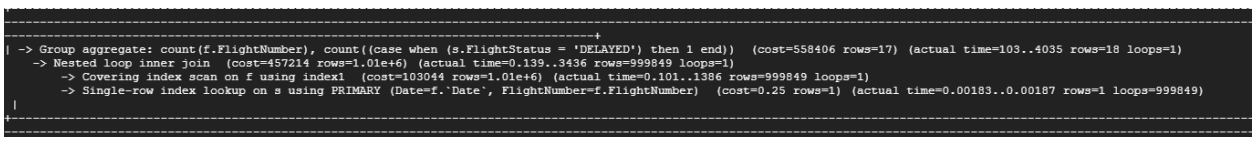


CREATE INDEX index1 ON Status (FlightStatus);
Cost: 0.25



Index on Status.FlightStatus did not improve the run cost because indexes are used to make row retrieval faster. However, FlightStatus was used with CASE WHEN, which doesn't retrieve rows.

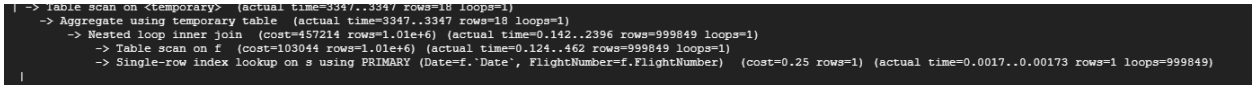
CREATE INDEX index1 ON Flight (AirlineName);
Cost: 0.25



Index on AirlineName did not improve the run cost because the data set is small and there are only 18 different airlines. Thus, it might prefer a full scan over using an index or the aggregation cost is negligible.

CREATE INDEX index1 ON Status(FlightNumber)

Cost: 0.25



This index did not reduce runtime because there is no index on the flight table. This forces the query to perform a full scan on the flight table.

Since none of the indexes affected the runtime, we chose to use no indexes.

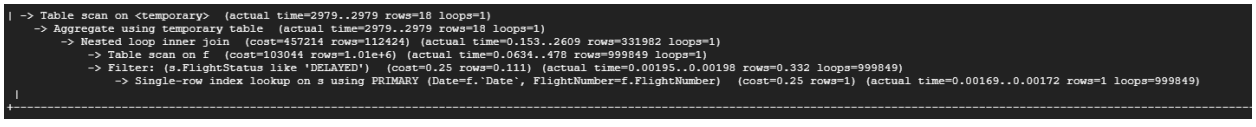
-- Average Late Time by Airline

```
SELECT late.name, AVG(late.Minutes_Late) AS Average_Minutes_Late
FROM (
   SELECT f.AirlineName AS name, s.FlightStatus, f.ActualDeptTime - f.DeptTime AS
Minutes_Late
   FROM Flight f
   JOIN Status s ON (s.Date = f.Date AND s.FlightNumber = f.FlightNumber)
   WHERE s.FlightStatus LIKE 'DELAYED'
) late
GROUP BY late.name;
```

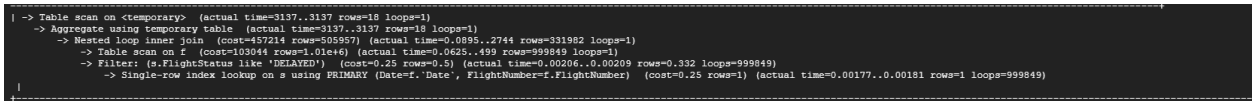
name	Average_Minutes_Late
JetBlue Airways	-8.0748
United Air Lines Inc	19.1094
Southwest Airlines C	28.4979
PSA Airlines Inc.	55.2207
SkyWest Airlines Inc	60.2094
Delta Air Lines Inc.	18.6352
American Airlines In	33.4617
Envoy Air	48.1245
Republic Airline	57.8078
Frontier Airlines In	-16.7892
Spirit Air Lines	6.2772
Alaska Airlines Inc.	24.1531
Mesa Airlines Inc.	54.6517
Hawaiian Airlines In	25.8017
ExpressJet Airlines	70.2998
Endeavor Air Inc.	55.3375
Allegiant Air	38.7061
Horizon Air	39.5105

18 rows in set (2.39 sec)

Explain analyze:
Cost: 0.25

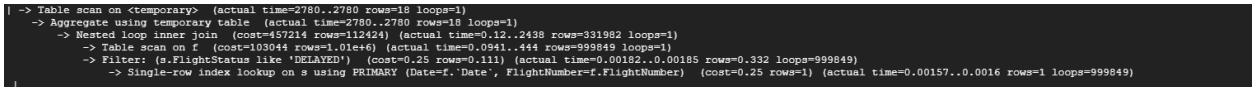


CREATE INDEX index1 ON Status (FlightStatus);
Cost: 0.25



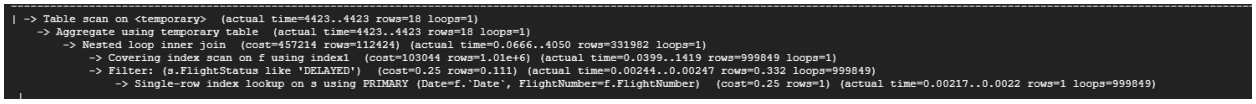
An index on FlightStatus didn't change the run cost could possibly be explained by there being too many delayed flights, which may cause the optimizer to prefer a full scan than to use the index.

CREATE INDEX index1 ON Flight (AirlineName);
Cost: 0.25



A reason that an index on AirlineName didn't affect run cost despite it being used in Group By can be possibly due to the data being too small. Thus the database may choose a hash aggregate.

CREATE INDEX index1 ON Flight (ActualDeptTime, AirlineName);
Cost: 0.25



Index on both ActualDeptTime and AirlineName didn't change the run cost because it doesn't include DeptTime since DeptTime is a Primary Key. Without DeptTime, the query will have to do a full scan, which makes the index ineffective.

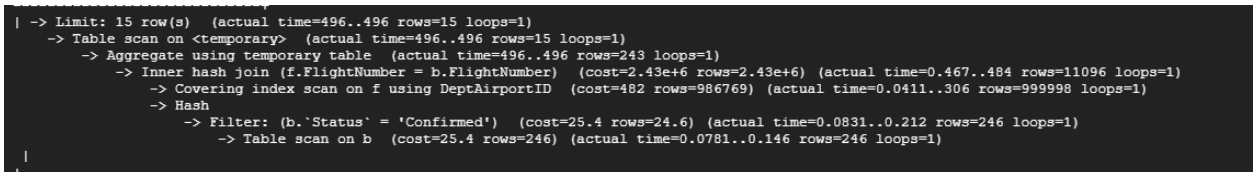
Since none of the indexes affected the runtime, we chose to use no indexes.

-- Flight Busyness
Select b.FlightNumber, Count(*)
From Booking b Join Flight f on (b.FlightNumber = f.FlightNumber)
Where b.status = 'Confirmed'
Group By b.FlightNumber;

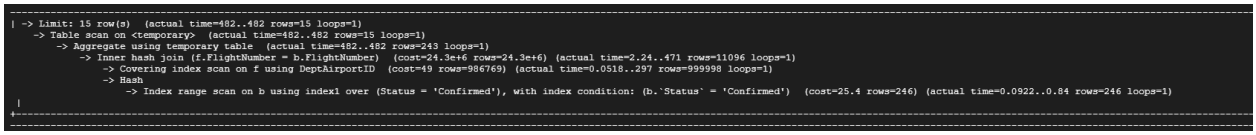
FlightNumber	Count(*)
WN 296	61
WN 527	60
DL 893	62
B6 1551	43
OO 3205	33
OO 4669	40
OO 3907	46
WN 727	69
MQ 3968	53
WN 756	53
DL 1200	63
WN 499	70
AS 1038	13
9E 4682	29
YV 5866	24

15 rows in set (0.72 sec)

Explain Anaylze
Cost: 25.4

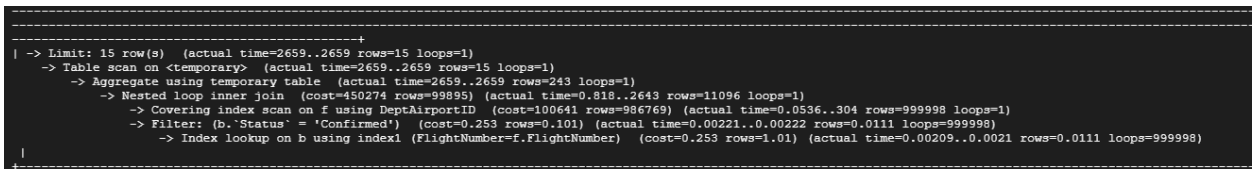


CREATE INDEX index1 ON Booking(status);
Cost: 25.4



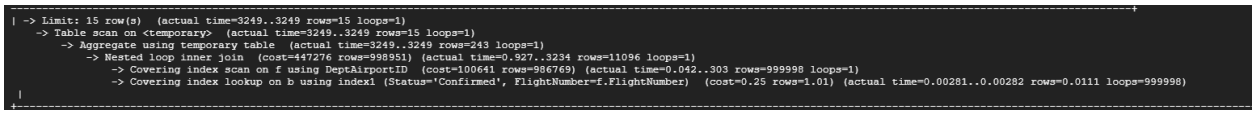
This index had no effect on the run cost. A reason for this could be due to a majority of the status attribute in the Booking table being ‘Confirmed,’ which causes the index to have little effect.

CREATE INDEX index1 ON Booking(FlightNumber);
Cost: 0.253



This index reduced the run cost from 25.4 to 0.253, which is a reduction by 25.14700.

CREATE INDEX index1 ON Booking(status, FlightNumber);
Cost: 0.25



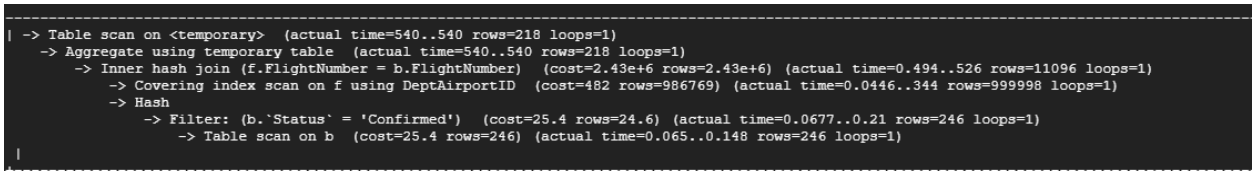
This index is the most effective, with it reducing the run cost from 25.4 to 0.25. Although it is only 0.003 better than the last one, it is still the lowest run cost. Thus, we will be using this index.

– How Early does Passengers book their flights per Departing Airport

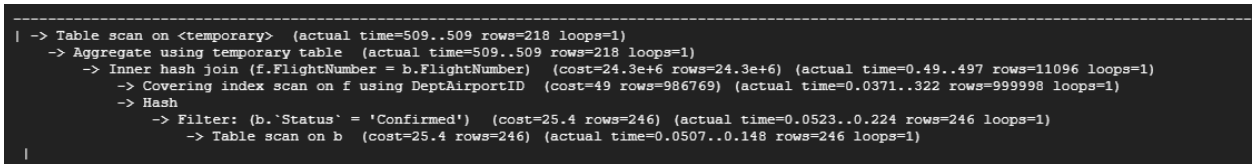
```
SELECT f.DeptAirportID, AVG(TIMESTAMPDIFF(YEAR, f.Date, b.BookingDate)) AS
AvgBookingAge
FROM Booking b
JOIN Flight f ON b.FlightNumber = f.FlightNumber
WHERE b.status = 'Confirmed'
GROUP BY f.DeptAirportID;
```

DeptAirportID	AvgBookingAge
ABQ	-1.2250
ALB	-0.2222
ACK	1.0000
ACV	-2.0000
ABE	-0.7500
ABI	0.0000
AEX	-0.7500
ACT	-2.0000
AMA	-2.5000
STL	-0.1957
SNA	0.1707
SMF	-0.9744
SLC	-0.5495
SEA	-1.0505
SAT	-1.2647

Explain Analyze:
Cost: 25.4

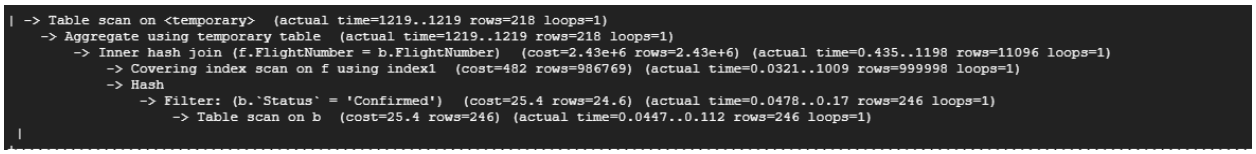


CREATE INDEX index1 ON Booking(status);
Cost 25.4



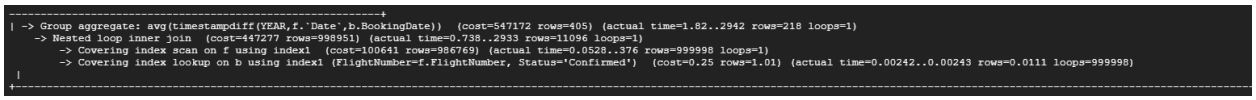
Although status is used in the WHERE clause so it should improve the run cost. However, the fact that it didn't change the run cost could be due to a majority of the status being 'Confirmed,' which causes the index to have little effect.

CREATE INDEX index1 ON Flight(DeptAirportID);
Cost: 25.4



This index didn't affect the run cost. A reason for this could be that there are very little unique values for AirportID so the index couldn't reduce the cost for aggregation.

CREATE INDEX index1 ON Booking (FlightNumber, status, BookingDate);
Cost: 0.25



This index had the best result with the run cost being reduced from 25.4 to 0.25. Thus, we will be using this index.