

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;

```

```

mysql> SELECT COUNT(*) FROM Status;
+-----+
| COUNT(*) |
+-----+
|    947464 |
+-----+
1 row in set (0.12 sec)

mysql> SELECT COUNT(*) FROM Flight;
+-----+
| COUNT(*) |
+-----+
|    947464 |
+-----+
1 row in set (0.28 sec)

mysql> SELECT COUNT(*) FROM Passenger;
+-----+
| COUNT(*) |
+-----+
|      1021 |
+-----+
1 row in set (0.00 sec)

mysql> █

```

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

```

-> Table scan on <temporary> (actual time=3092..3092 rows=18 loops=1)
-> Aggregate using temporary table (actual time=3092..3092 rows=18 loops=1)
-> Nested loop inner join (cost=457214 rows=1.01e+6) (actual time=0.116..2230 rows=999849 loops=1)
-> Table scan on f (cost=103044 rows=1.01e+6) (actual time=0.0973..415 rows=999849 loops=1)
-> Single-row index lookup on s using PRIMARY (Date=f.Date, FlightNumber=f.FlightNumber) (cost=0.25 rows=1) (actual time=0.00159..0.00162 rows=1 loops=999849)

```

CREATE INDEX index1 ON Status (FlightStatus);

Cost: 0.25

```

-> Table scan on <temporary> (actual time=3196..3196 rows=18 loops=1)
-> Aggregate using temporary table (actual time=3196..3196 rows=18 loops=1)
-> Nested loop inner join (cost=457214 rows=1.01e+6) (actual time=0.178..2278 rows=999849 loops=1)
-> Table scan on f (cost=103044 rows=1.01e+6) (actual time=0.15..426 rows=999849 loops=1)
-> Single-row index lookup on s using PRIMARY (Date=f.Date, FlightNumber=f.FlightNumber) (cost=0.25 rows=1) (actual time=0.00163..0.00166 rows=1 loops=999849)

```

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

```

-> Group aggregate: count(f.FlightNumber), count((case when (s.FlightStatus = 'DELAYED') then 1 end)) (cost=558406 rows=17) (actual time=103..4035 rows=18 loops=1)
-> Nested loop inner join (cost=457214 rows=1.01e+6) (actual time=0.139..3436 rows=999849 loops=1)
-> Covering index scan on f using index1 (cost=103044 rows=1.01e+6) (actual time=0.101..1386 rows=999849 loops=1)
-> Single-row index lookup on s using PRIMARY (Date=f.Date, FlightNumber=f.FlightNumber) (cost=0.25 rows=1) (actual time=0.00183..0.00187 rows=1 loops=999849)

```

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

```
| -> Table scan on <temporary> (actual time=3347.3347 rows=18 loops=1)
|   -> Aggregate using temporary table (actual time=3347.3347 rows=18 loops=1)
|     -> Nested loop inner join (cost=457214 rows=1.01e+6) (actual time=0.142..2396 rows=999849 loops=1)
|       -> Table scan on f (cost=103044 rows=1.01e+6) (actual time=0.124..462 rows=999849 loops=1)
|       -> Single-row index lookup on s using PRIMARY (Date=f.'Date', FlightNumber=f.FlightNumber) (cost=0.25 rows=1) (actual time=0.0017..0.00173 rows=1 loops=999849)
```

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

```
| -> Table scan on <temporary> (actual time=2979.2979 rows=18 loops=1)
|   -> Aggregate using temporary table (actual time=2979.2979 rows=18 loops=1)
|     -> Nested loop inner join (cost=457214 rows=112424) (actual time=0.153..2609 rows=331982 loops=1)
|       -> Table scan on f (cost=103044 rows=1.01e+6) (actual time=0.0634..478 rows=999849 loops=1)
|       -> Filter: (s.FlightStatus like 'DELAYED') (cost=0.25 rows=0.111) (actual time=0.00195..0.00198 rows=0.332 loops=999849)
|       -> Single-row index lookup on s using PRIMARY (Date=f.'Date', FlightNumber=f.FlightNumber) (cost=0.25 rows=1) (actual time=0.00169..0.00172 rows=1 loops=999849)
```

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

```
| -> Table scan on <temporary> (actual time=3137..3137 rows=18 loops=1)
|   -> Aggregate using temporary table (actual time=3137..3137 rows=18 loops=1)
|     -> Nested loop inner join (cost=457214 rows=503957) (actual time=0.0895..2744 rows=331982 loops=1)
|       -> Table scan on f (cost=103044 rows=1.01e+6) (actual time=0.0625..499 rows=999849 loops=1)
|       -> Filter: (s.FlightStatus like 'DELAYED') (cost=0.25 rows=0.5) (actual time=0.00206..0.00209 rows=0.332 loops=999849)
|         -> Single-row index lookup on s using PRIMARY (Date=f.Date, FlightNumber=f.FlightNumber) (cost=0.25 rows=1) (actual time=0.00177..0.00181 rows=1 loops=999849)
```

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

```
| -> Table scan on <temporary> (actual time=2780..2780 rows=18 loops=1)
|   -> Aggregate using temporary table (actual time=2780..2780 rows=18 loops=1)
|     -> Nested loop inner join (cost=457214 rows=112424) (actual time=0.12..2438 rows=331982 loops=1)
|       -> Table scan on f (cost=103044 rows=1.01e+6) (actual time=0.0941..444 rows=999849 loops=1)
|       -> Filter: (s.FlightStatus like 'DELAYED') (cost=0.25 rows=0.111) (actual time=0.00182..0.00185 rows=0.332 loops=999849)
|         -> Single-row index lookup on s using PRIMARY (Date=f.Date, FlightNumber=f.FlightNumber) (cost=0.25 rows=1) (actual time=0.00157..0.0016 rows=1 loops=999849)
```

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

```
| -> Table scan on <temporary> (actual time=4423..4423 rows=18 loops=1)
|   -> Aggregate using temporary table (actual time=4423..4423 rows=18 loops=1)
|     -> Nested loop inner join (cost=457214 rows=112424) (actual time=0.0666..4050 rows=331982 loops=1)
|       -> Covering index scan on f using index1 (cost=103044 rows=1.01e+6) (actual time=0.0399..1415 rows=999849 loops=1)
|       -> Filter: (s.FlightStatus like 'DELAYED') (cost=0.25 rows=0.111) (actual time=0.00244..0.00247 rows=0.332 loops=999849)
|         -> Single-row index lookup on s using PRIMARY (Date=f.Date, FlightNumber=f.FlightNumber) (cost=0.25 rows=1) (actual time=0.00217..0.0022 rows=1 loops=999849)
```

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 Analyze

Cost: 25.4

```

| -> Limit: 15 row(s) (actual time=496.496 rows=15 loops=1)
      -> Table scan on <temporary> (actual time=496.496 rows=15 loops=1)
            -> Aggregate using temporary table (actual time=496.496 rows=243 loops=1)
                  -> Inner hash join (f.FlightNumber = b.FlightNumber) (cost=2.43e+6 rows=2.43e+6) (actual time=0.467..484 rows=11096 loops=1)
                        -> Covering index scan on f using DeptAirportID (cost=482 rows=986769) (actual time=0.0411..306 rows=999998 loops=1)
                        -> Hash
                              -> Filter: (b.`Status` = 'Confirmed') (cost=25.4 rows=24.6) (actual time=0.0831..0.212 rows=246 loops=1)
                                      -> Table scan on b (cost=25.4 rows=246) (actual time=0.0781..0.146 rows=246 loops=1)

```

CREATE INDEX index1 ON Booking(status);

Cost: 25.4

```

| -> Limit: 15 row(s) (actual time=482.482 rows=15 loops=1)
      -> Table scan on <temporary> (actual time=482.482 rows=15 loops=1)
            -> Aggregate using temporary table (actual time=482.482 rows=243 loops=1)
                  -> Inner hash join (f.FlightNumber = b.FlightNumber) (cost=24.3e+6 rows=24.3e+6) (actual time=2.24..471 rows=11096 loops=1)
                        -> Covering index scan on f using DeptAirportID (cost=49 rows=986769) (actual time=0.0518..297 rows=999998 loops=1)
                        -> Hash
                              -> Index range scan on b using index1 over (Status = 'Confirmed'), with index condition: (b.`Status` = 'Confirmed') (cost=25.4 rows=246) (actual time=0.0922..0.84 rows=246 loops=1)

```

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

```

| -> Limit: 15 row(s) (actual time=2659.2659 rows=15 loops=1)
      -> Table scan on <temporary> (actual time=2659.2659 rows=15 loops=1)
            -> Aggregate using temporary table (actual time=2659.2659 rows=243 loops=1)
                  -> Nested loop inner join (cost=450274 rows=99895) (actual time=0.818..2643 rows=11096 loops=1)
                        -> Covering index scan on f using DeptAirportID (cost=100641 rows=986769) (actual time=0.0536..304 rows=999998 loops=1)
                        -> Filter: (b.`Status` = 'Confirmed') (cost=0.253 rows=0.101) (actual time=0.00221..0.00222 rows=0.0111 loops=999998)
                              -> Index lookup on b using index1 (FlightNumber=f.FlightNumber) (cost=0.253 rows=1.01) (actual time=0.00209..0.0021 rows=0.0111 loops=999998)

```

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

```

| -> Limit: 15 row(s) (actual time=3249..3249 rows=15 loops=1)
| -> Table scan on <temporary> (actual time=3249..3249 rows=15 loops=1)
| -> Aggregate using temporary table (actual time=3249..3249 rows=243 loops=1)
| -> Nested loop inner join (cost=447276 rows=998951) (actual time=0.927..3234 rows=11096 loops=1)
| -> Covering index scan on f using DeptAirportID (cost=100641 rows=986769) (actual time=0.042..303 rows=999998 loops=1)
| -> Covering index lookup on b using indexl (Status='Confirmed', FlightNumber=f.FlightNumber) (cost=0.25 rows=1.01) (actual time=0.00281..0.00282 rows=0.0111 loops=999998)
|

```

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

```

| -> Table scan on <temporary> (actual time=540..540 rows=218 loops=1)
| -> Aggregate using temporary table (actual time=540..540 rows=218 loops=1)
| -> Inner hash join (f.FlightNumber = b.FlightNumber) (cost=2.43e+6 rows=2.43e+6) (actual time=0.494..526 rows=11096 loops=1)
| -> Covering index scan on f using DeptAirportID (cost=482 rows=986769) (actual time=0.0446..344 rows=999998 loops=1)
| -> Hash
| -> Filter: (b.'Status' = 'Confirmed') (cost=25.4 rows=24.6) (actual time=0.0677..0.21 rows=246 loops=1)
| -> Table scan on b (cost=25.4 rows=246) (actual time=0.065..0.148 rows=246 loops=1)
|

```

CREATE INDEX index1 ON Booking(status);

Cost 25.4

```

-----+-----
| -> Table scan on <temporary> (actual time=509..509 rows=218 loops=1)
|   -> Aggregate using temporary table (actual time=509..509 rows=218 loops=1)
|     -> Inner hash join (f.FlightNumber = b.FlightNumber) (cost=24.3e+6 rows=24.3e+6) (actual time=0.49..497 rows=11096 loops=1)
|       -> Covering index scan on f using DeptAirportID (cost=49 rows=986769) (actual time=0.0371..322 rows=999998 loops=1)
|       -> Hash
|         -> Filter: (b.'Status' = 'Confirmed') (cost=25.4 rows=246) (actual time=0.0523..0.224 rows=246 loops=1)
|         -> Table scan on b (cost=25.4 rows=246) (actual time=0.0507..0.148 rows=246 loops=1)
|
+-----+-----

```

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

```

-----+-----
| -> Table scan on <temporary> (actual time=1219..1219 rows=218 loops=1)
|   -> Aggregate using temporary table (actual time=1219..1219 rows=218 loops=1)
|     -> Inner hash join (f.FlightNumber = b.FlightNumber) (cost=2.43e+6 rows=2.43e+6) (actual time=0.435..1198 rows=11096 loops=1)
|       -> Covering index scan on f using index1 (cost=482 rows=986769) (actual time=0.0321..1009 rows=999998 loops=1)
|       -> Hash
|         -> Filter: (b.'Status' = 'Confirmed') (cost=25.4 rows=24.6) (actual time=0.0478..0.17 rows=246 loops=1)
|         -> Table scan on b (cost=25.4 rows=246) (actual time=0.0447..0.112 rows=246 loops=1)
|
+-----+-----

```

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

```

-----+-----
| -> Group aggregate: avg(timestampdiff(YEAR,f.'Date',b.BookingDate)) (cost=547172 rows=485) (actual time=1.82..2942 rows=218 loops=1)
|   -> Nested loop inner join (cost=447277 rows=998991) (actual time=0.738..2933 rows=11096 loops=1)
|     -> Covering index scan on f using index1 (cost=100641 rows=986769) (actual time=0.0528..376 rows=999998 loops=1)
|     -> Covering index lookup on b using index1 (FlightNumber=f.FlightNumber, Status='Confirmed') (cost=0.25 rows=1.01) (actual time=0.00242..0.00243 rows=0.0111 loops=999998)
|
+-----+-----

```

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.