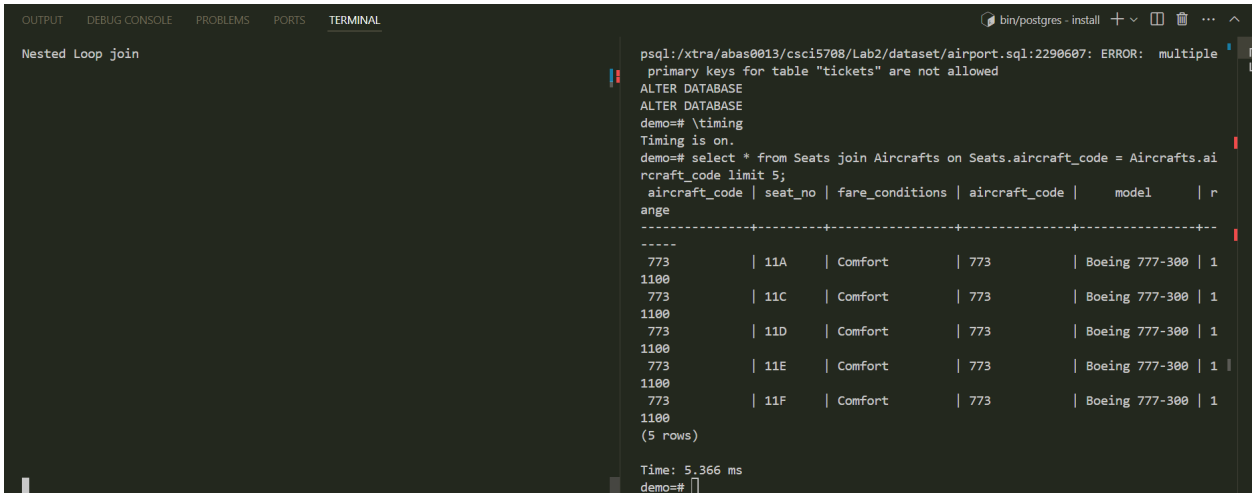
Christopher Miao
Sulekha Abas

A.

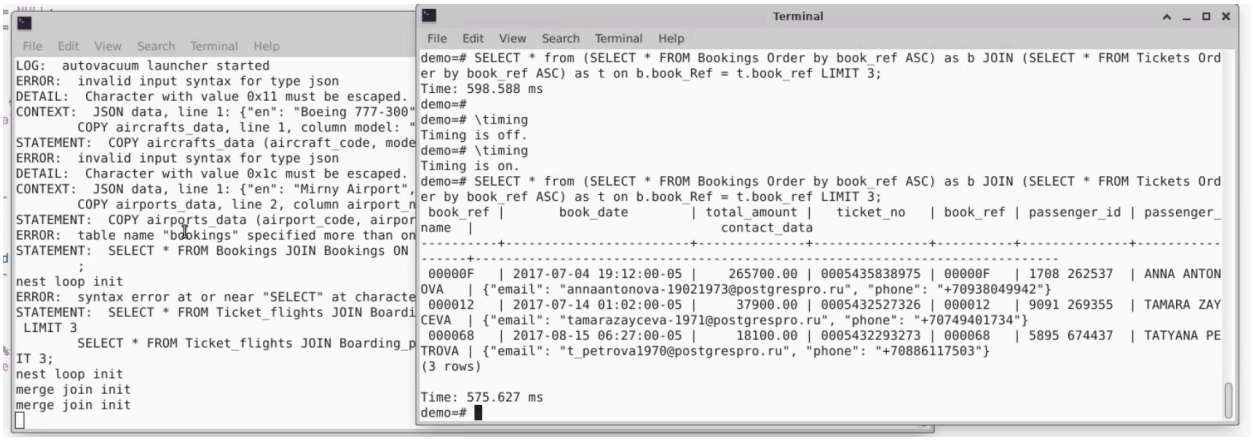
This is our nested loop join showing that and the query performance is time of 5.366 ms.

NESTED LOOP JOIN: Q1



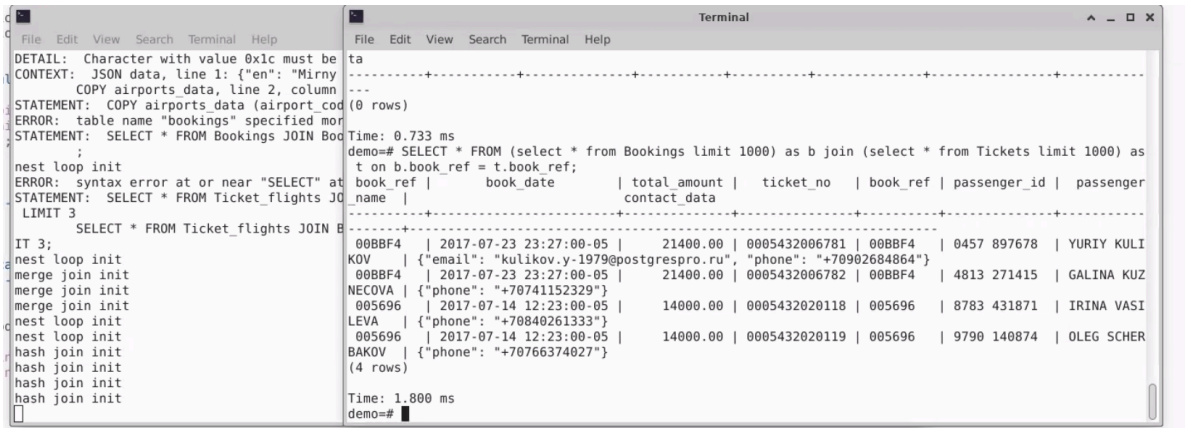
This is our merge join with query performance of 575.627 ms.

MERGE JOIN: Q2



This is our hash join with query performance of 1.800 ms.

HASH JOIN: Q3



B.

Below is Query 1, which originally resulted in a Nested Loop join. When run using a nested loop join, it took 0.590 ms. This nested loop join query did not work when we forced it to use a merge join. We tried several other queries that were originally shown to be nested loop joins, but when we tried making those other queries use a merge join, they all would not run. We also tried it on two different machines and faced the same issues. Therefore, we do not know what the exact runtime of an originally nested loop join query would be if it were forced to use a merge join instead. However, when our original nested loop join query was forced to run as a hash merge, it worked fine and had a run time of 0.692 ms.

```
Nest
demo=# select * from Seats join Aircrafts on Seats.aircraft_code = Aircrafts.aircraft_code limit 5;
 aircraft_code | seat_no | fare_conditions | aircraft_code | model | range
---------------+---------+-----------------+---------------+-------+-------
 773           | 11A     | Comfort         | 773           |       | 11100
 773           | 11C     | Comfort         | 773           |       | 11100
 773           | 11D     | Comfort         | 773           |       | 11100
 773           | 11E     | Comfort         | 773           |       | 11100
 773           | 11F     | Comfort         | 773           |       | 11100
(5 rows)

Time: 0.590 ms
demo=#
```

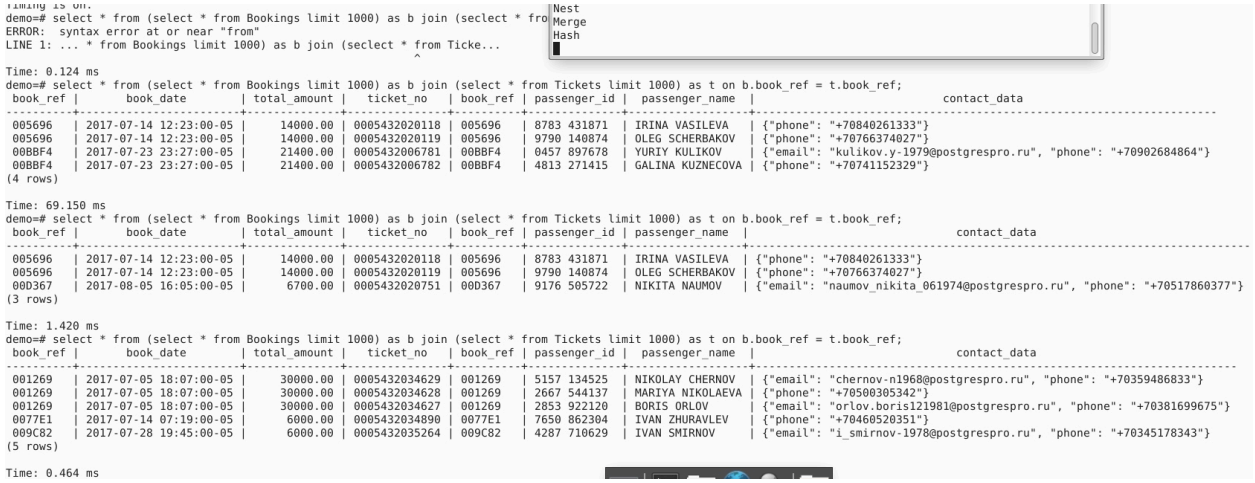
```
Hash
demo=# select * from Seats join Aircrafts on Seats.aircraft_code = Aircrafts.aircraft_code limit 5;
 aircraft_code | seat_no | fare_conditions | aircraft_code | model | range
---------------+---------+-----------------+---------------+-------+-------
 319           | 2A      | Business        | 319           |       |  6700
 319           | 2C      | Business        | 319           |       |  6700
 319           | 2D      | Business        | 319           |       |  6700
 319           | 2F      | Business        | 319           |       |  6700
 319           | 3A      | Business        | 319           |       |  6700
(5 rows)

Time: 0.692 ms
demo=#
```

Below is Query 2, which originally resulted in a Merge join. The time it took for Q2 to run using a nested loop join was 1062.6 ms. When Q2 used a merge join, it took 640.1 ms, and when it used a Hash 1024.3 ms.

```
demo=# select * from (select * from Bookings order by book_ref asc) as b join (select * from Tickets order by book_ref asc) as t on b.book_ref = t. book_ref limit 3;
 book_ref |       book_date        | total_amount |   ticket_no   | book_ref | passenger_id | passenger_name  |                           contact_data
----------+------------------------+--------------+---------------+----------+--------------+-----------------+-----------------------------------------------------------------
 00000F   | 2017-07-04 19:12:00-05 |    265700.00 | 0005435838975 | 00000F   | 1708 262537  | ANNA ANTONOVA   | {"email": "annaantonova-19021973@postgrespro.ru", "phone": "+70938049942"}
 000012   | 2017-07-14 01:02:00-05 |     37900.00 | 0005432527326 | 000012   | 9091 269355  | TAMARA ZAYCEVA  | {"email": "tamarazayceva-1971@postgrespro.ru", "phone": "+70749401734"}
 000068   | 2017-08-15 06:27:00-05 |     18100.00 | 0005432293273 | 000068   | 5895 674437  | TATYANA PETROVA | {"email": "t_petrova1970@postgrespro.ru", "phone": "+70886117503"}
(3 rows)

Time: 1062.593 ms
demo=# select * from (select * from Bookings order by book_ref asc) as b join (select * from Tickets order by book_ref asc) as t on b.book_ref = t. book_ref limit 3;
 book_ref |       book_date        | total_amount |   ticket_no   | book_ref | passenger_id | passenger_name  |                           contact_data
----------+------------------------+--------------+---------------+----------+--------------+-----------------+-----------------------------------------------------------------
 00000F   | 2017-07-04 19:12:00-05 |    265700.00 | 0005435838975 | 00000F   | 1708 262537  | ANNA ANTONOVA   | {"email": "annaantonova-19021973@postgrespro.ru", "phone": "+70938049942"}
 000012   | 2017-07-14 01:02:00-05 |     37900.00 | 0005432527326 | 000012   | 9091 269355  | TAMARA ZAYCEVA  | {"email": "tamarazayceva-1971@postgrespro.ru", "phone": "+70749401734"}
 000068   | 2017-08-15 06:27:00-05 |     18100.00 | 0005432293273 | 000068   | 5895 674437  | TATYANA PETROVA | {"email": "t_petrova1970@postgrespro.ru", "phone": "+70886117503"}
(3 rows)

Time: 640.145 ms
demo=# select * from (select * from Bookings order by book_ref asc) as b join (select * from Tickets order by book_ref asc) as t on b.book_ref = t. book_ref limit 3;
 book_ref |       book_date        | total_amount |   ticket_no   | book_ref | passenger_id |  passenger_name  |                           contact_data
----------+------------------------+--------------+---------------+----------+--------------+------------------+-----------------------------------------------------------------
 000068   | 2017-08-15 06:27:00-05 |     18100.00 | 0005432293273 | 000068   | 5895 674437  | TATYANA PETROVA  | {"email": "t_petrova1970@postgrespro.ru", "phone": "+70886117503"}
 0006C3   | 2017-08-02 03:42:00-05 |     59400.00 | 0005432329491 | 0006C3   | 4486 459735  | YULIYA TROFIMOVA | {"email": "yuliyatrofimova-011988@postgrespro.ru", "phone": "+70850747734"}
 0006C3   | 2017-08-02 03:42:00-05 |     59400.00 | 0005432329492 | 0006C3   | 1691 126633  | FARIT ZAKHAROV   | {"email": "zakharov-farit27091972@postgrespro.ru", "phone": "+70301259324"}
(3 rows)

Time: 1024.300 ms
```

```
nest loop init
merge join init
hash join init
```

Below is Query 3, which originally resulted in a Hash join. The time it took for Q3 to run as a nested loop join was 69.150 ms, with a merge join it took 1.420 ms, and with a hash join it took 0.464 ms.



```
demo=# select * from (select * from Bookings limit 1000) as b join (seclect * fro
ERROR:  syntax error at or near "from"
LINE 1: ... * from Bookings limit 1000) as b join (seclect * from Ticke...
                                                   ^

Time: 0.124 ms
demo=# select * from (select * from Bookings limit 1000) as b join (select * from Tickets limit 1000) as t on b.book_ref = t.book_ref;
 book_ref |       book_date        | total_amount |   ticket_no   | book_ref | passenger_id |  passenger_name  |                        contact_data
----------+------------------------+--------------+---------------+----------+--------------+------------------+-----------------------------------------------------------
 005696   | 2017-07-14 12:23:00-05 |     14000.00 | 0005432020118 | 005696   | 8783 431871  | IRINA VASILEVA   | {"phone": "+70840261333"}
 005696   | 2017-07-14 12:23:00-05 |     14000.00 | 0005432020119 | 005696   | 9790 140874  | OLEG SCHERBAKOV  | {"phone": "+70766374027"}
 00BBF4   | 2017-07-23 23:27:00-05 |     21400.00 | 0005432006781 | 00BBF4   | 0457 897678  | YURIY KULIKOV    | {"email": "kulikov.y-1979@postgrespro.ru", "phone": "+70902684864"}
 00BBF4   | 2017-07-23 23:27:00-05 |     21400.00 | 0005432006782 | 00BBF4   | 4813 271415  | GALINA KUZNECOVA | {"phone": "+70741152329"}
(4 rows)

Time: 69.150 ms
demo=# select * from (select * from Bookings limit 1000) as b join (select * from Tickets limit 1000) as t on b.book_ref = t.book_ref;
 book_ref |       book_date        | total_amount |   ticket_no   | book_ref | passenger_id | passenger_name  |                        contact_data
----------+------------------------+--------------+---------------+----------+--------------+-----------------+-----------------------------------------------------------
 005696   | 2017-07-14 12:23:00-05 |     14000.00 | 0005432020118 | 005696   | 8783 431871  | IRINA VASILEVA  | {"phone": "+70840261333"}
 005696   | 2017-07-14 12:23:00-05 |     14000.00 | 0005432020119 | 005696   | 9790 140874  | OLEG SCHERBAKOV | {"phone": "+70766374027"}
 00D367   | 2017-08-05 16:05:00-05 |      6700.00 | 0005432020751 | 00D367   | 9176 505722  | NIKITA NAUMOV   | {"email": "naumov_nikita_061974@postgrespro.ru", "phone": "+70517860377"}
(3 rows)

Time: 1.420 ms
demo=# select * from (select * from Bookings limit 1000) as b join (select * from Tickets limit 1000) as t on b.book_ref = t.book_ref;
 book_ref |       book_date        | total_amount |   ticket_no   | book_ref | passenger_id |  passenger_name  |                        contact_data
----------+------------------------+--------------+---------------+----------+--------------+------------------+-----------------------------------------------------------
 001269   | 2017-07-05 18:07:00-05 |     30000.00 | 0005432034629 | 001269   | 5157 134525  | NIKOLAY CHERNOV  | {"email": "chernov-n1968@postgrespro.ru", "phone": "+70359486833"}
 001269   | 2017-07-05 18:07:00-05 |     30000.00 | 0005432034628 | 001269   | 2667 544137  | MARIYA NIKOLAEVA | {"phone": "+70500305342"}
 001269   | 2017-07-05 18:07:00-05 |     30000.00 | 0005432034627 | 001269   | 2853 922120  | BORIS ORLOV      | {"email": "orlov.boris121981@postgrespro.ru", "phone": "+70381699675"}
 0077E1   | 2017-07-14 07:19:00-05 |      6000.00 | 0005432034890 | 0077E1   | 7650 862304  | IVAN ZHURAVLEV   | {"phone": "+70460520351"}
 009C82   | 2017-07-28 19:45:00-05 |      6000.00 | 0005432035264 | 009C82   | 4287 710629  | IVAN SMIRNOV     | {"email": "i_smirnov-1978@postgrespro.ru", "phone": "+70345178343"}
(5 rows)

Time: 0.464 ms
```

C. For the original nested loop, the best query for this should be the nest loop, because there is neither sorting nor does it fit well into memory. But the nested loop would not run for merge join even when we tried to force it to choose that join method. We tried to change the computer that we were running it on and changed the dataset to be in the same exports file and tried to do a fresh install of the lab. But none of these allowed us to run our nestloop query as a merge join. But between the two joins that ran, nested loop and hash join, nested loop performed better. We went to office hours and the TA said that this would be fine for the submission.

For the original merge join, the best query is merge join since it has the shortest runtime of 640.1 ms compared to nest loop which was 1062.6 ms and hash join which was 1024.3 ms.

For the original hash join, the best query is hash join since it has the shortest runtime of 0.464 ms compared to merge join which took 1.420 ms and nest loop which took 69.150 ms.

D.
For the original nested loop, we think that the nest loop join algorithm is better because we chose two relatively large unsorted tables to join together, so that hash join and merge join would not be favored, so the only result would be a simple scan and nest loop would be favored.

For the original merge join, we think that the merge join algorithm is the best since we made our query sort each table before running the overall join, resulting in the query favoring merge join.

For the original hash join, we think that the hash join algorithm is the best since we made our query limit each table before joining, this way the tables would fit within memory and hash join would be favored.

E.
For the original nested loop, it does match the original query optimizer that was run in step 1.

For the original merge join, it does match the original query optimizer that was run in step 1.

For the original hash join, it does match the original query optimizer that was run in step 1.

F.
For the original nested loop, we think that the original postgresql took the right decision for this query. Since our query is neither sorted nor small enough to fit in memory.

For the original merge join, we think that the original postgresql took the right decision for this query. Since our query is sorted, merge join should be favored.

For the original hash join, we think that the original postgresql took the right decision. We can say this because when Q3 (the originally hash joined query) was run using the other types of joins (the nested loop and merge joins) the query had a longer run time and was fastest when it used a hash join.