Middle East Technical University | Department of Computer Engineering

CENG 352

Database Management Systems

Spring 2019-2020
Project 1 : SQL Queries

Due date: February 23, 2020, Sunday, 23:59

1 Objectives

In this project you are asked to write several SQL queries on a relational flight database. The data in this database is from **U.S. Department of Transformation**, which is officially released public subset of flight data that is available for access to everyone.

Your SQL server will be PostgreSQL server and you will work on your local environment. You can see tutorials about how to setup working environment for PostgreSQL.

2 Database Schema

2.1 Tables

```
-> airline_codes ( [airline_code], airline_name )
-> airport_codes ( [airport_code], airport_desc )
-> airport_ids ( [airport_id], airport_desc )
-> cancellation_reasons ( [reason_code], reason_desc )
-> weekdays ( [weekday_id], weekday_name )
-> world_area_codes ( [wac_id], wac_name )
-> flight_reports ( [report_id], year, month, day, weekday_id, airline_code,
plane_tail_number, origin_airport_id, origin_airport_code, origin_city_name,
origin_wac_id, dest_airport_id, dest_airport_code, dest_city_name, dest_wac_id,
departure_time, departure_delay, taxi_out_time, wheels_off_time, wheels_on_time,
taxi_in_time, arrival_time, arrival_delay, is_cancelled, cancellation_reason,
is_diverted, flight_time, flight_distance )
```

2.2 Explanations of Columns

- **weekday_id:** A number between 1-7, which represents days from Monday to Sunday.

- **airline_code:** A short string that represents the airline in flight_reports entries. Length is mostly between 2-6.
- **plane_tail_number:** A string that identifies a plane. It is unique to the plane itself, but not the airline. You may see reports with same plane_tail_number but different airline_code. This is because the plane is sold to another airline at some time.
- **origin_airport_id & dest_airport_id:** Numbers that represent airports in flight_reports entries. Used for identifying the airport in origin city and in destination city.
- **origin_airport_code & dest_airport_code:** Strings that represent airports in flight_reports entries. Used for identifying the airport in origin city and in destination city.
- **origin_city_name & dest_city_name:** Origin and destination city names.
- **origin_wac_id & dest_wac_id:** Numbers that represent world area codes in flight_reports entries. Used for identifying the state in countries. In data, you can see cities like San Francisco or Denver. However, San Francisco belongs to California world area (state) and Denver belongs to Colorado world area.
- **departure_time:** Time that the plane **starts** to leave the gate in the airport. It is in HH:MM format.
- **departure_delay:** Difference in minutes between scheduled and actual departure time. Early departures show negative numbers.
- **taxi_out_time:** Time spent by a plane between the gate and the takeoff. Planes leave the gate, pass the taxiway, take their position on the runway and take off in taxi_out_time minutes.
- **wheels_off_time:** Time that the plane takes off and becomes airborne. It is in HH:MM format. You can consider:

  ```
  departure_time + taxi_out_time = wheels_off_time
  ```

- **wheels_on_time:** Time that the plane touches the ground in the destination airport. It is in HH:MM format.
- **taxi_in_time:** Time spent by a plane between the touchdown and the parking to the gate. Planes touch down to runway, pass the taxiway, take their parking position in the gate in taxi_in_time minutes.
- **arrival_time:** Time that the plane stops at the gate in the destination airport. It is in HH:MM format. You can consider:

  ```
  wheels_on_time + taxi_in_time = arrival_time
  ```

- **arrival_delay:** Difference in minutes between scheduled and actual arrival time. Early arrivals show negative numbers.
- **is_cancelled:** Indicator for cancelled flight. This field is either 1 or 0. When a flight is cancelled, there is a cancellation_reason for that.

- **cancellation_reason:** A character that specifies the reason for cancellation. If flight is not cancelled (is_cancelled = 0), this field is NULL. Otherwise, the reason can be 'A' (Carrier), 'B' (Weather), 'C' (National Air System), 'D' (Security).
- **is_diverted:** A diverted flight is one that has been routed from its original arrival destination to a new, typically temporary, arrival destination. This field is either 1 or 0.
- **flight_time:** Time spent by a plane in the air in minutes.
- **flight_distance:** Distance between origin and destination airports in miles.

2.3 Foreign Keys

Here, in table flight_reports,

- **weekday_id** is a foreign key that references to weekday_id in **weekdays** table
- **airline_code** is a foreign key that references to airline_code in **airline_codes** table
- **origin_airport_id, dest_airport_id** are foreign keys that reference to airline_id in **airline_ids** table
- **origin_airport_code, dest_airport_code** are foreign keys that reference to airport_code in **airline_codes** table
- **origin_wac_id, dest_wac_id** are foreign keys that reference to wac_id in **world_area_codes** table
- **cancellation_reason** is a foreign key that references to reason_code in **cancellation_reasons** table

You can check **this link** for more understanding of the data.

3 SQL Queries

1. Find airlines and their average departure delay in 2018. Show airline name in ascending order, airline code and average departure delay in ascending order. Remember to disregard cancelled flights. (18 rows)

```
Columns:  [airline_name *] [airline_code] [avg_delay *]
           ...               ...            ...
           ...               ...            ...

* avg_delay -> ascending order
* airline_name -> ascending order (for equal avg_delay values)
```

2. List airports and the number of cancelled flights with 'Security' reason. Show airport code, airport description and cancelled flight count in descending order. (53 rows)

```
Columns:  [airport_code *] [airport_desc] [cancel_count *]
           ...              ...               ...
           ...              ...               ...

* cancel_count -> descending order
* airport_code -> ascending order (for equal cancel_count values)
```

***CLARIFICATION:** You should only consider **origin airport code** for this query.

3. List planes that received maintenance at the end of the year. When a plane has more than 5 flights per day in a yearly period, it receives maintenance. Show plane tail number, year, daily flight average. Remember to disregard cancelled flights. (3821 rows)

```
Columns:  [plane_tail_number *] [year *] [daily_avg]
           ...                   ...       ...
           ...                   ...       ...

* plane_tail_number -> ascending order
* year -> ascending order (for equal plane_tail_number values)
```

***CLARIFICATION:** You need to find the number of non-cancelled flights of planes for each day. Then you need to check yearly flight count averages considering daily flight counts that you previously found. If the yearly average is more than 5, then the plane went under maintenance.

4. Find airlines that goes to **ALL of** 'Boston, MA', 'New York, NY', 'Portland, ME', 'Washington, DC', 'Philadelphia, PA' cities, **in 2018 or 2019**. You should disregard the ones that already flying these cities in 2016 or 2017. You should also ignore cancelled flights. Show airline names only. (5 rows)

```
Columns:  [airline_name *]
           ...
           ...

* airline_name -> ascending order
```

5. Find all non-cancelled travels from Seattle to Boston with one stop. Flights should happen in the same day (flight1: Seattle => Destination, flight2: Destination => Boston). Order flights by their total time, in ascending order.

```
Total time = flight1(flight_time) + flight1(taxi_out_time)
           + flight2(taxi_in_time) + flight2(flight_time)

Total distance = flight1(flight_distance) + flight2(flight_distance)
```

Remember to check that **flight1(arrival_time) < flight2(departure_time)**. Show flight date as "DD/MM/YYYY" string, plane tail number, flight1 arrival_time, flight2 departure_time, flight1 origin_city_name, stop_city_name, flight2 dest_city_name, total_time, total_distance. (253 rows)

```
Columns: flight_date*, plane_tail_number*, flight1_arrival_time,
flight2_departure_time, origin_city_name, stop_city_name*, dest_city_name,
total_time*, total_distance*

    * total_time -> ascending order
    * total_distance -> ascending order
    * plane_tail_number -> ascending order
    * stop_city_name -> ascending order
```

6. Find best weekday for flights from San Francisco to Boston. Best weekday is the day that has least "departure_delay + arrival_delay" daily average. Show weekday_id, weekday_name, average delay. (1 row only, the best one)

```
Columns:  [weekday_id] [weekday_name] [avg_delay]
           ...              ...           ...  (only 1 row)
```

7. Find all airlines that had more than 10% of their flights out of Boston are cancelled. Return the airline name and the percentage of canceled flights out of Boston. Order the results by the percentage of canceled flights in descending order. (2 rows)

```
Columns:  [airline_name] [percentage*]
           ...            ...
           ...            ...

* percentage -> descending order
```

8. Sometimes an airline can buy planes from another airline and re-brand it. Find sold and re-branded planes. Display plane tail number, first owner airline code and second owner airline code. You need to check same plane tail number for different airlines.

   Airline A can use plane X in 2016 and airline B can use same plane in 2018, 2019. Therefore ('X', 'A', 'B') should be in the query result. (189 rows)

```
Columns:  [plane_tail_number *] [first_owner] [second_owner]
           ...                   ...           ...
           ...                   ...           ...

* plane_tail_number -> ascending order
```

   ***CLARIFICATION:** If same planes are used by different airlines for different dates, then it is re-branded. For different dates, date1 < date2:

   - if year1 < year2
   - or if year1 = year2 and month1 < month2
   - or if year1 < year2 and month1 = month2 and day1 < day2

9. Find average speed of planes that flew **ONLY** weekends of January 2016. Show plane tail number and average speed. Results should have an descending order on average speed. (15 rows)

```
Columns:  [plane_tail_number] [avg_speed*]
           ...                 ...
           ...                 ...

* avg_speed -> descending order
```

10. Find airlines that have planes that **ONLY** goes to Texas area. Display airline name and the number of flights by those planes. (2 rows)

```
Columns:  [airline_name *] [flight_count]
           ...              ...
           ...              ...

* airline_name -> ascending order
```

11. List the popular airline names, total number of flights, total number of cancelled flights for each year. Popular airlines are airlines that have more than 2000 flights in a day on the average, and have this popularity in all years. (12 rows)

```
Columns:  [airline_name *] [year *] [total_num_flights] [cancelled_flights]
           ...              ...          ...               ...
           ...              ...          ...               ...

* airline_name -> ascending order
```

***CLARIFICATION:** To find popular airlines, you need to look at daily flight counts, and take the average of those counts. If the average is above 2000 flights per day, for all years, then the airline is a popular airline.

12. For each year and airline codes, find number of flights to Boston and the percentage of Boston flights to the overall number of flights of that airline in that year. Consider results with the percentage $> 1\%$. Disregard cancelled flights. (28 rows)

```
[year *] [airline_code *] [boston_flight_count] [boston_flight_percentage]
  ...            ...             ...                    ...
  ...            ...             ...                    ...

* year -> ascending order
* airline_code -> ascending order
```

13. For each airline, list the total number of flights on Monday and the total number of flights on Sunday in separate columns. Disregard cancelled flights.

```
[airline_name *] [monday_flights] [sunday_flights]
  ...                 ...             ...
  ...                 ...             ...

* airline_name -> ascending order
```

14. For each year and weekday, find the frequent cancellation reason and number of such cancellations.

```
[year *] [weekday_name] [reason] [number_of_cancellations]
  ...       ...           ...             ...
  ...       ...           ...             ...

* year -> ascending order
```

15. Find top 5 airports with highest flight traffic. Disregard cancelled flights. (5 rows)
Total traffic is defined as:

```
Traffic = Outgoing flight count + Incoming flight count

or

Traffic = Flight count where the airport is the origin airport
           + Flight count where the airport is the destination airport

    Columns:  [airport_desc *]
               ...
               ...

    * airport_desc -> ascending order
```

4 Regulations

1. **Submission:** Submission will be done via ODTUClass. Please remember that **late submission is allowed 5 days for all programming proejcts.** You can distribute these 5 days to any mini-project your want. You should put the answer query to each question in a separate .sql file and zip those .sql files with following name:

```
e1234567_project1.zip
    -> q1.sql
    -> q2.sql
    ...
```

Where you should replace "1234567" with your own student number.

2. **SQL Style:** You should write your queries in readable manner. Write each clause on a separate line, add indentation for clear look. For example:

   This is easier to read

```
select
  *
from
  flight_reports fr
where
  fr.origin_city_name like '%Denver%'
  and fr.is_cancelled = 0
```

   than this

```
select * from flight_reports fr where fr.origin_city_name like '%Denver%'
and fr.is_cancelled = 0
```

   You can use online-formatters/beautifiers for better indentation if you want.

3. **Newsgroup:** You must follow the newsgroup (news.ceng.metu.edu.tr) for discussions and possible updates on a daily basis.

5 Tutorial & Guides

- You can download PostgreSQL server from **here**.
- For visualization, **DBeaver** is a nice tool which works for both Ubuntu and Windows. You can also use it for many other database servers.
- Once you start database server, open DBeaver and click 'New Database Connection' on top left corner. (Figure 1)
- Choose PostgreSQL. (Figure 2)
- Enter credentials and connect to default 'postgres' database. You will create your own database for this project later. (Figure 3)
- Open a new script and execute **"create database ceng352_flight_data"** by selecting script and hitting (CTRL + ENTER). (Figure 4)
- Now that you have created another database, connect to newly created database just like before. (Figure 5)
- Open 'Tables' to see the tables. (Figure 6) Since there are no tables you need to create tables using the prepared script. Download it from ODTUClass and open it with DBeaver software. Run queries by selecting parts of script and hitting CTRL + ENTER. (Figure 7)
- You are almost there. Now, you need to import the data to tables. Import data other than "flight_reports_YYYY.sql"s, leave them to the end for foreign keys.

- To import data, right click on the table name and click 'Import data' (Figure 8)
- Choose CSV type and .csv file that you want to import for selected table. (Figure 9, 10)
- ***IMPORTANT NOTE*** Choose **semicolon(;)** as delimiter for 'flight_reports_YYYY.csv' files. Choose **comma (,)** as delimiter for other .csv files.
- That's it. Now you can write queries on tables.

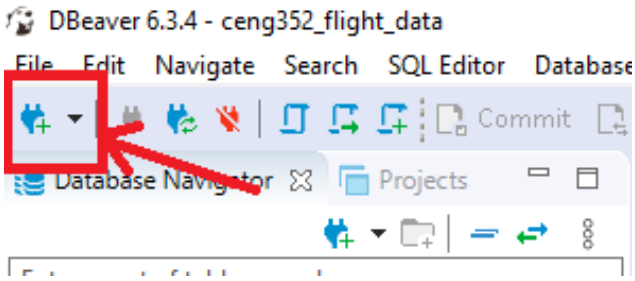

Figure 1: New database connection

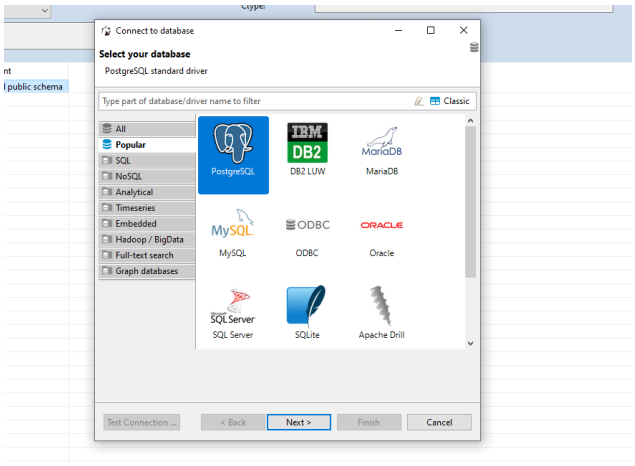

Figure 2: Database options

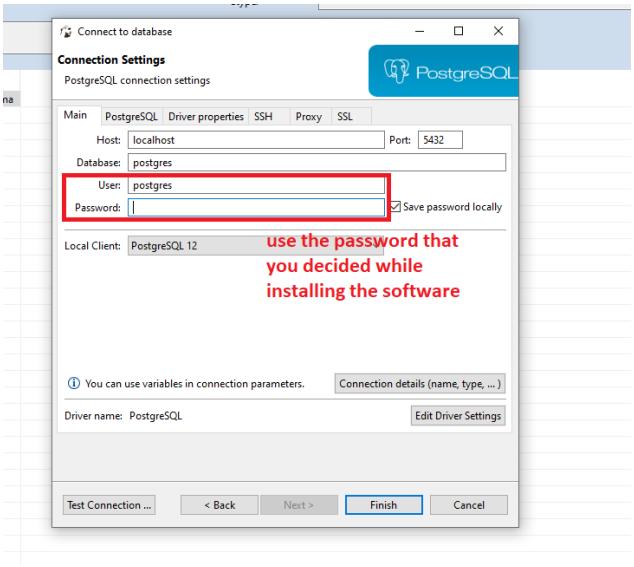

Figure 3: Enter credentials and connect

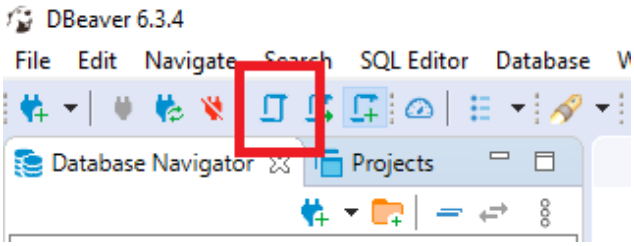

Figure 4: Open new script

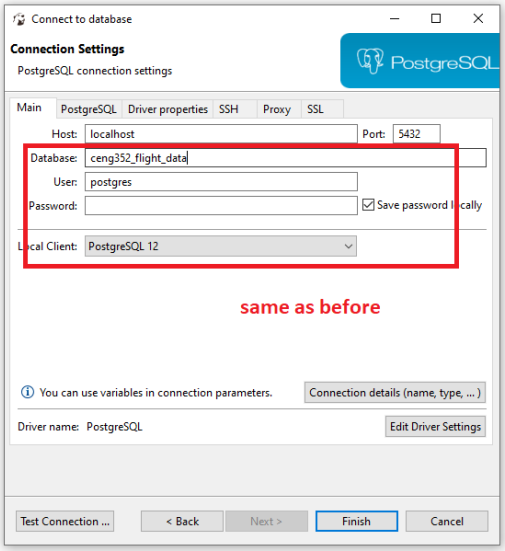

Figure 5: Connect to new database

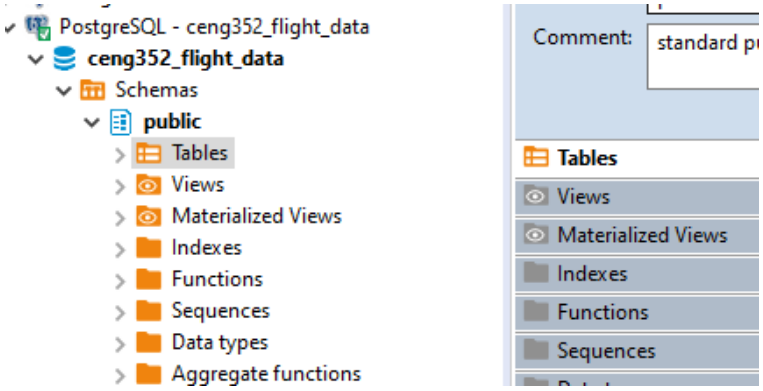

Figure 6: Open tables view

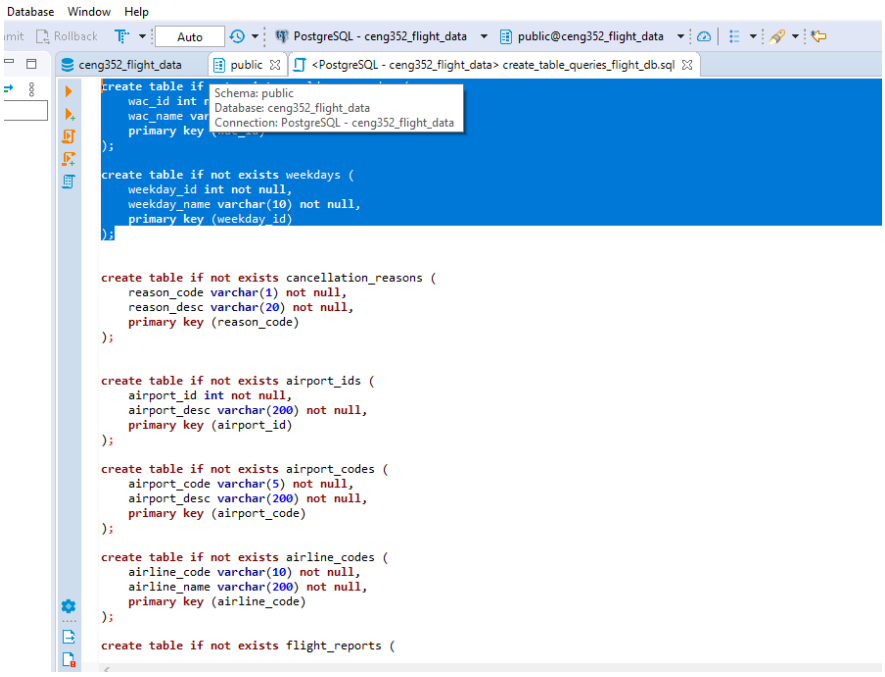

Figure 7: Run create scripts

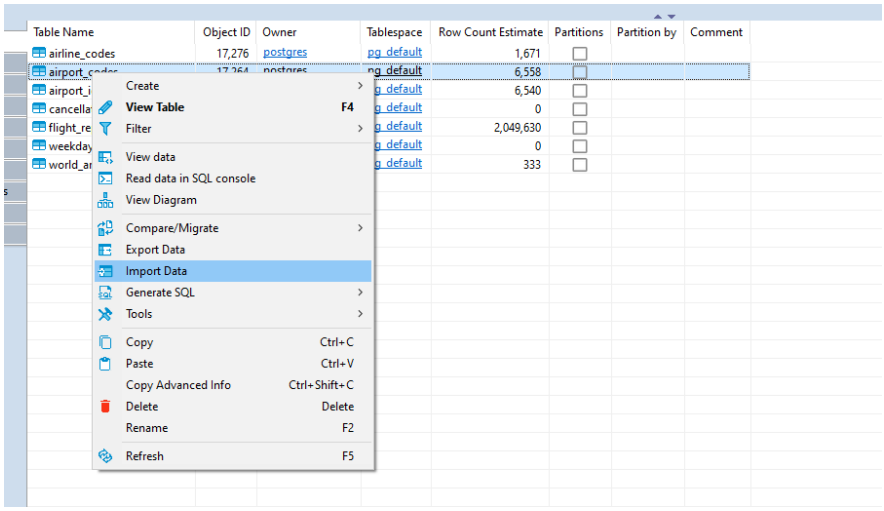

Figure 8: Import data

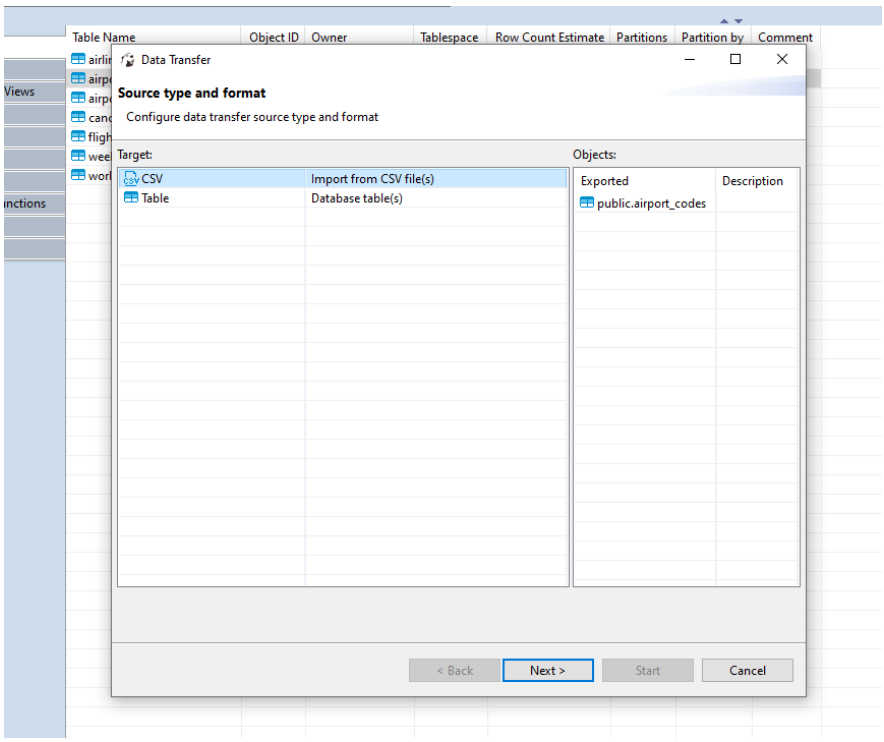

Figure 9: Choose CSV

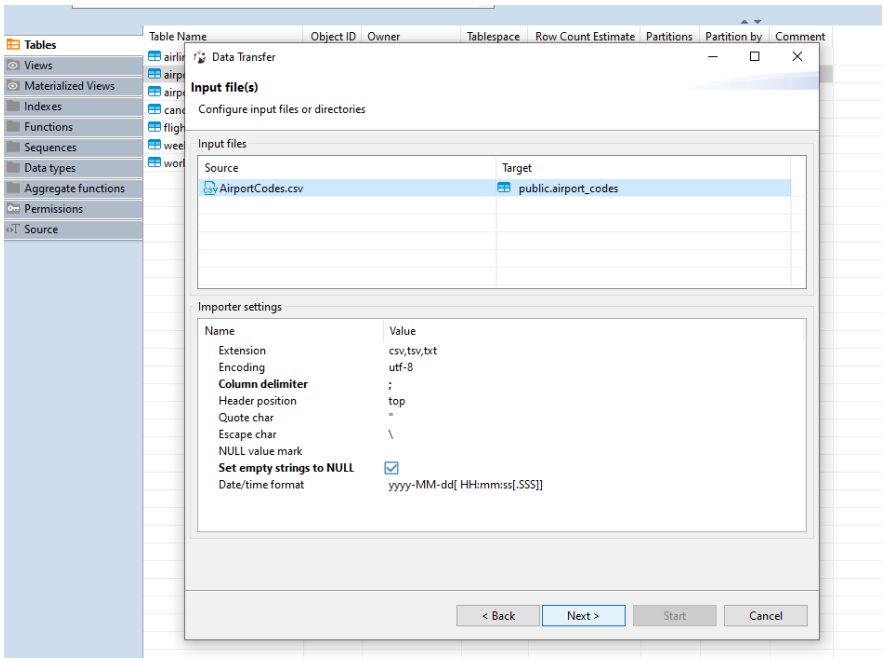

Figure 10: Set delimiter