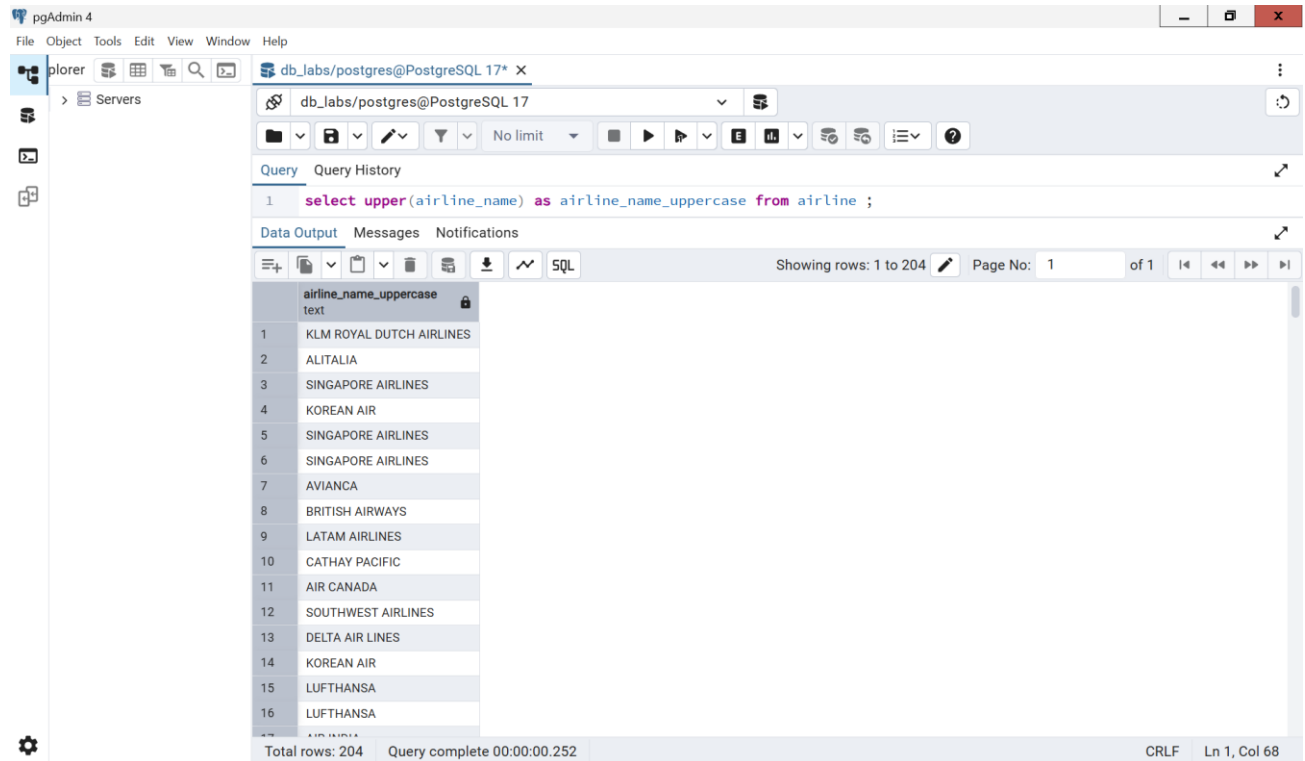


Lab4

1. Retrieve all airline names in uppercase.

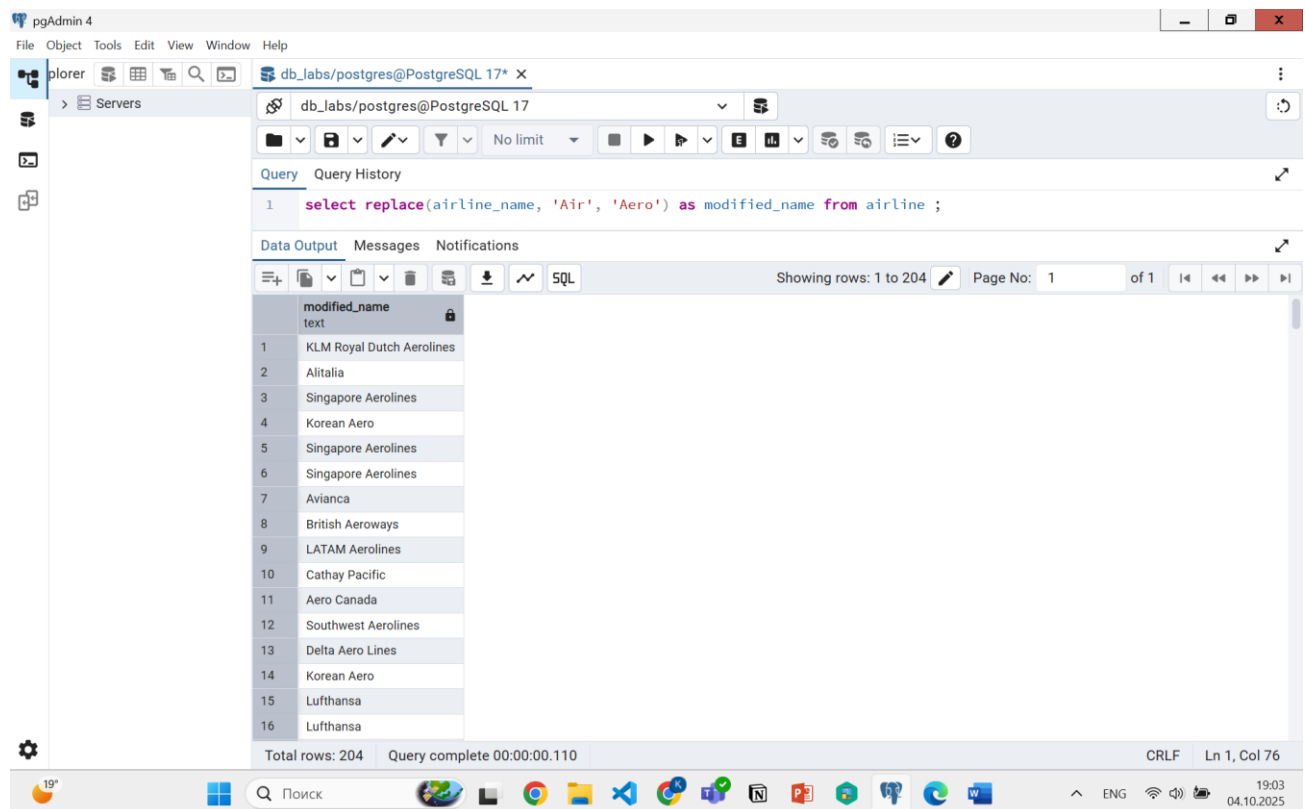


The screenshot shows the pgAdmin 4 interface. The left sidebar displays the 'Servers' tree. The main pane shows a query window for 'db_labs/postgres@PostgreSQL 17*'. The query is: `select upper(airline_name) as airline_name_uppercase from airline ;`. The 'Data Output' tab is active, showing a table with 204 rows. The first 16 rows are visible, showing the airline names in uppercase.

airline_name_uppercase
1 KLM ROYAL DUTCH AIRLINES
2 ALITALIA
3 SINGAPORE AIRLINES
4 KOREAN AIR
5 SINGAPORE AIRLINES
6 SINGAPORE AIRLINES
7 AVIANCA
8 BRITISH AIRWAYS
9 LATAM AIRLINES
10 CATHAY PACIFIC
11 AIR CANADA
12 SOUTHWEST AIRLINES
13 DELTA AIR LINES
14 KOREAN AIR
15 LUFTHANSA
16 LUFTHANSA

Total rows: 204 Query complete 00:00:00.252 CRLF Ln 1, Col 68

2. Replace any occurrence of the word "Air" in airline names with "Aero".

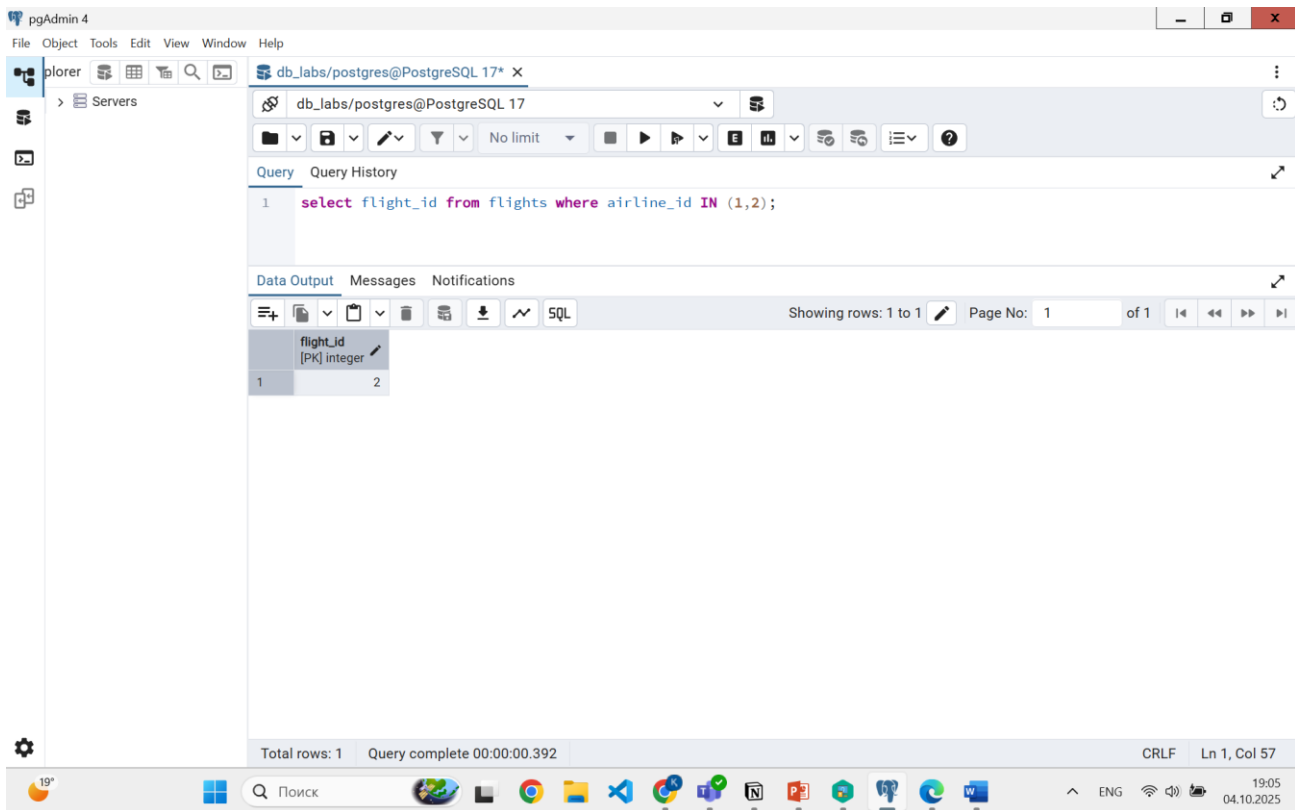


The screenshot shows the pgAdmin 4 interface. The left sidebar displays the 'Servers' tree. The main pane shows a query window for 'db_labs/postgres@PostgreSQL 17*'. The query is: `select replace(airline_name, 'Air', 'Aero') as modified_name from airline ;`. The 'Data Output' tab is active, showing a table with 204 rows. The first 16 rows are visible, showing the modified airline names with 'Aero' instead of 'Air'.

modified_name
1 KLM Royal Dutch Aerolines
2 Alitalia
3 Singapore Aerolines
4 Korean Aero
5 Singapore Aerolines
6 Singapore Aerolines
7 Avianca
8 British Aeroways
9 LATAM Aerolines
10 Cathay Pacific
11 Aero Canada
12 Southwest Aerolines
13 Delta Aero Lines
14 Korean Aero
15 Lufthansa
16 Lufthansa

Total rows: 204 Query complete 00:00:00.110 CRLF Ln 1, Col 76

3. Find all flight numbers that coordinates with both airline 1 and airline 2.



The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL query:

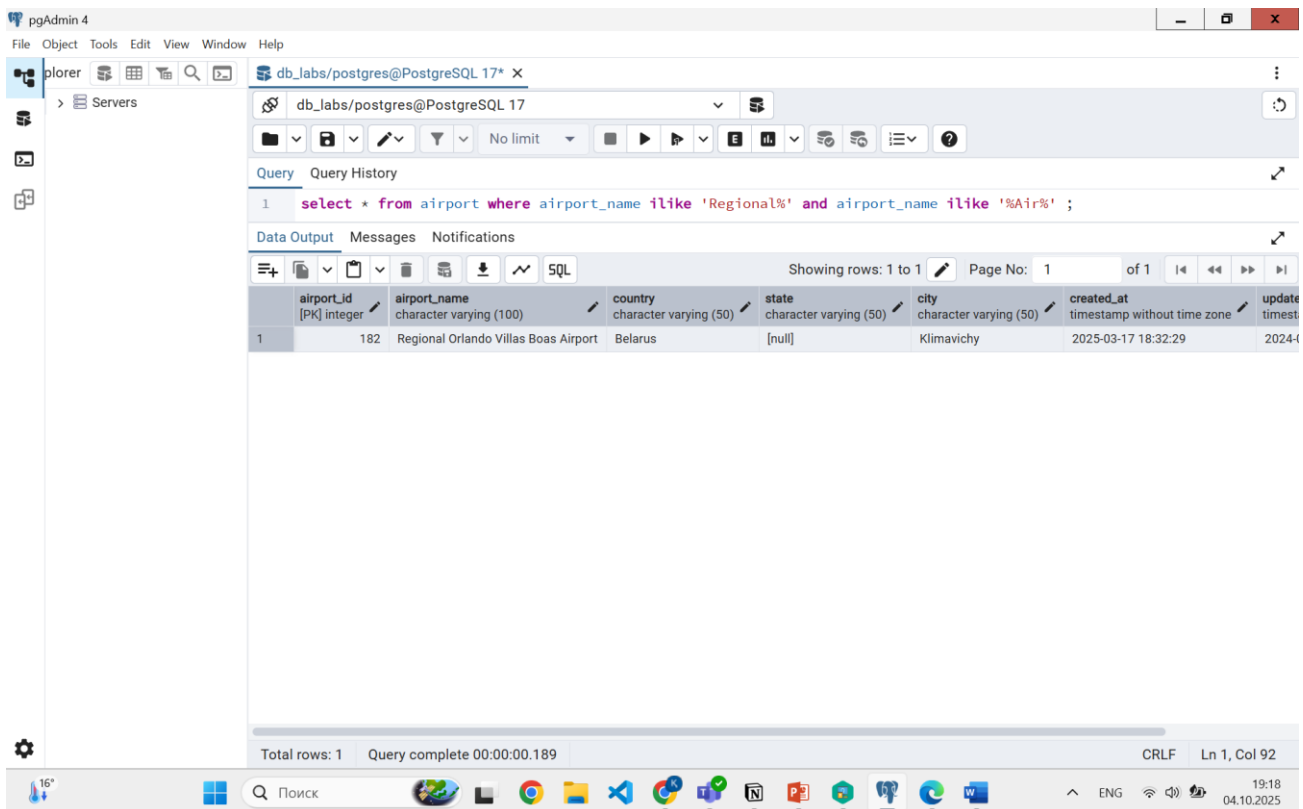
```
1 select flight_id from flights where airline_id IN (1,2);
```

The Data Output tab shows the results of the query:

flight_id [PK] integer
1
2

The status bar at the bottom indicates "Total rows: 1" and "Query complete 00:00:00.392".

4. Retrieve airports that contain the word "Reginal" and "Air" in their names.



The screenshot shows the pgAdmin 4 interface. The query editor contains the following SQL query:

```
1 select * from airport where airport_name ilike 'Regional%' and airport_name ilike '%Air%';
```

The Data Output tab shows the results of the query:

airport_id [PK] integer	airport_name character varying (100)	country character varying (50)	state character varying (50)	city character varying (50)	created_at timestamp without time zone	update timestamp
1	182 Regional Orlando Villas Boas Airport	Belarus	[null]	Klimavichy	2025-03-17 18:32:29	2024-4

The status bar at the bottom indicates "Total rows: 1" and "Query complete 00:00:00.189".

5. Retrieve passenger names and format their birth dates as 'Month DD, YYYY'..o

The screenshot shows the pgAdmin 4 interface with a SQL query executed against the 'passengers' table. The query formats the birth date into 'Month DD, YYYY' format. The results are displayed in a table with 16 rows.

```
1 select first_name, last_name, to_char(date_of_birth, 'Month DD, YYYY') as formatted_birth_date from passengers ;
```

	first_name character varying (50)	last_name character varying (50)	formatted_birth_date text
1	Jacqui	Rous	February 17, 2025
2	Averill	Machin	May 02, 2025
3	Rosalind	Critoph	February 07, 2025
4	Torey	Towe	April 23, 2025
5	Tye	Dulling	August 10, 2025
6	Fabio	Nutbeem	June 15, 2025
7	Carlie	Millions	December 27, 2024
8	Tara	Whannel	March 26, 2025
9	Obadiah	Acome	April 16, 2025
10	Binky	Figg	May 05, 2025
11	Karalynn	Boler	May 19, 2025
12	Abrahan	Blunkett	January 09, 2025
13	Heath	Manach	December 18, 2024
14	Cornell	Yacobsohn	December 31, 2024
15	Alonso	Iacapucci	October 28, 2024
16	Zulema	Toomer	April 16, 2025

Total rows: 200 Query complete 00:00:00.175

6. Find flight numbers that have been delayed based on the actual arrival time.

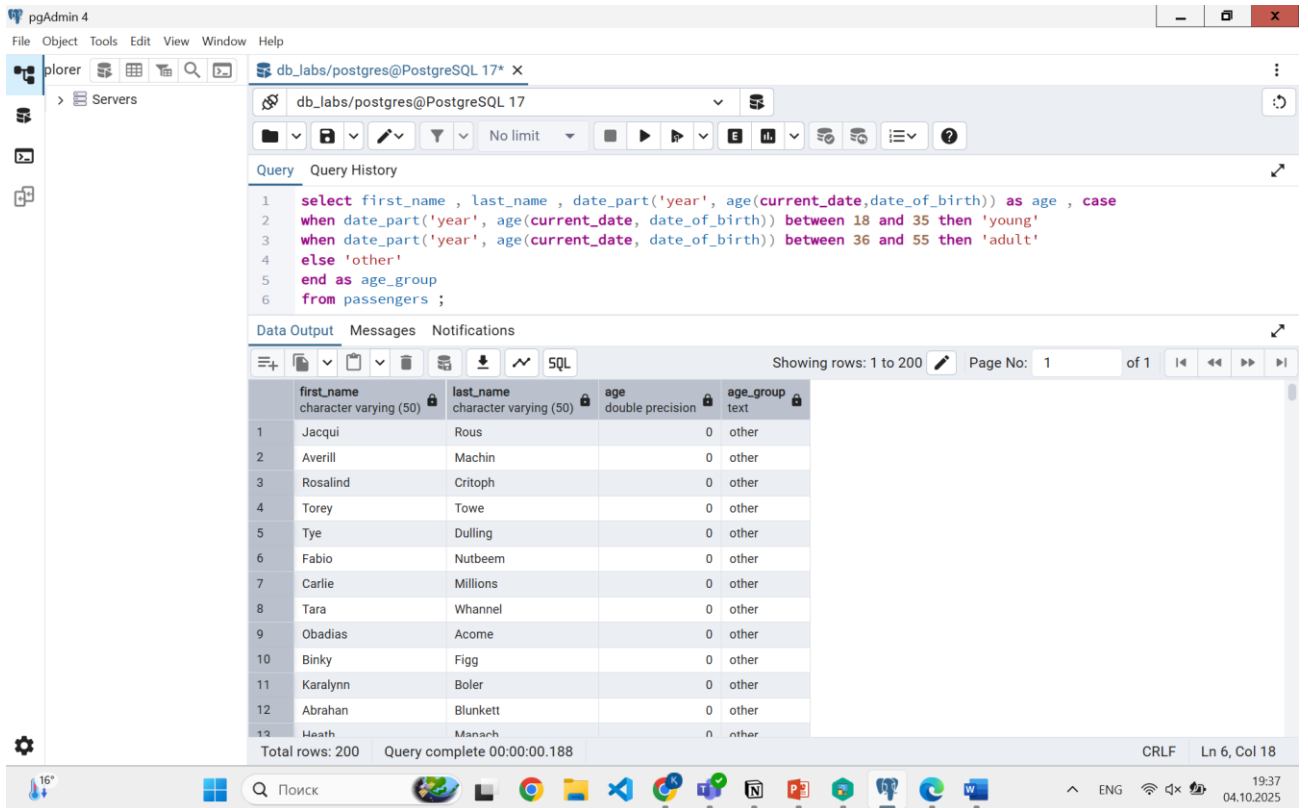
The screenshot shows the pgAdmin 4 interface with a SQL query executed against the 'flights' table. The query identifies flights where the actual arrival time is greater than the scheduled arrival time. The results are displayed in a table with 16 rows.

```
1 select flight_id from flights where act_arrival_time > sch_arrival_time;
```

	flight_id [PK] integer
1	4
2	6
3	11
4	21
5	22
6	26
7	27
8	29
9	32
10	34
11	36
12	37
13	40
14	41
15	42
16	43

Total rows: 64 Query complete 00:00:00.159

7. Create a query that divides passengers into age groups like 'Young' and 'Adult' based on their birth date. Young passengers age between 18 and 35, Adult passengers age between 36 and 55.



The screenshot shows the pgAdmin 4 interface. The Query tab displays the following SQL query:

```

1 select first_name , last_name , date_part('year', age(current_date,date_of_birth)) as age , case
2 when date_part('year', age(current_date, date_of_birth)) between 18 and 35 then 'young'
3 when date_part('year', age(current_date, date_of_birth)) between 36 and 55 then 'adult'
4 else 'other'
5 end as age_group
6 from passengers ;

```

The Data Output tab shows the results of the query. The table has four columns: first_name, last_name, age, and age_group. The results are as follows:

	first_name	last_name	age	age_group
1	Jacqui	Rous	0	other
2	Averill	Machin	0	other
3	Rosalind	Critoph	0	other
4	Torey	Towe	0	other
5	Tye	Dulling	0	other
6	Fabio	Nutbeem	0	other
7	Carlie	Millions	0	other
8	Tara	Whannel	0	other
9	Obadiah	Acome	0	other
10	Binky	Figg	0	other
11	Karalynn	Boler	0	other
12	Abrahan	Blunkett	0	other
13	Heath	Mananah	0	other

Total rows: 200 Query complete 00:00:00.188

8. Create a query that categorizes ticket prices based on their price as "Cheap," "Medium" or "Expensive."

pgAdmin 4

File Object Tools Edit View Window Help

db_labs/postgres@PostgreSQL 17*

db_labs/postgres@PostgreSQL 17

Query

```
1 select booking_id, ticket_price, case
2 when ticket_price < 100 then 'cheap'
3 when ticket_price between 100 and 300 then 'medium'
4 else 'expensive'
5 end as price_category
6 from booking;
```

Data Output Messages Notifications

Showing rows: 1 to 24 Page No: 1 of 1

	booking_id [PK] integer	ticket_price numeric (7,2)	price_category text
1	2	68.99	cheap
2	22	57.49	cheap
3	41	172.49	medium
4	44	91.99	cheap
5	55	228.85	medium
6	56	57.49	cheap
7	63	68.99	cheap
8	70	51.74	cheap
9	95	149.49	medium
10	73	413.99	expensive
11	106	229.99	medium
12	111	91.99	cheap

Total rows: 24 Query complete 00:00:00.171 CRLF Ln 3, Col 52

9. Find number of airline names in each airline country.

pgAdmin 4

File Object Tools Edit View Window Help

db_labs/postgres@PostgreSQL 17*

db_labs/postgres@PostgreSQL 17

Query

```
1 select airline_country, count(*) as number_of_airlines
2 from airline
3 group by airline_country
4 order by number_of_airlines desc;
```

Data Output Messages Notifications

Showing rows: 1 to 64 Page No: 1 of 1

	airline_country character varying (50)	number_of_airlines bigint
1	China	42
2	Indonesia	20
3	Philippines	11
4	France	9
5	Russia	8
6	Greece	7
7	Argentina	6
8	United States	6
9	Brazil	6
10	Portugal	6
11	Poland	5
12	Colombia	5
13	Sweden	4
14	Thailand	3

Total rows: 64 Query complete 00:00:00.155 CRLF Ln 3, Col 26

10. Find flights that arrived late according to their actual arrival time compared to the scheduled arrival time.

The screenshot shows the pgAdmin 4 interface. The query editor displays the following SQL query:

```
1 select flight_id, sch_arrival_time, act_arrival_time
2 from flights
3 where act_arrival_time > sch_arrival_time ;
```

The query results are displayed in a table with the following columns: `flight_id` (integer), `sch_arrival_time` (timestamp without time zone), and `act_arrival_time` (timestamp without time zone). The table shows 64 rows of data, with the first 15 rows visible in the screenshot. The status bar at the bottom indicates "Total rows: 64" and "Query complete 00:00:00.304".

	flight_id	sch_arrival_time	act_arrival_time
1	4	2025-01-21 11:53:36	2025-02-18 19:31:23
2	6	2025-03-15 16:22:18	2025-06-24 05:30:54
3	11	2025-04-09 01:20:33	2025-05-11 21:13:38
4	21	2025-02-05 17:00:51	2025-02-24 19:41:29
5	22	2025-02-10 15:05:11	2025-04-23 14:15:40
6	26	2025-02-03 11:38:53	2025-04-11 15:06:05
7	27	2025-05-06 19:34:57	2025-08-04 02:16:36
8	29	2025-02-05 21:26:16	2025-05-11 23:07:54
9	32	2025-03-22 21:09:22	2025-08-08 16:03:14
10	34	2025-01-20 08:13:53	2025-04-05 07:20:53
11	36	2025-01-07 15:15:30	2025-08-27 09:34:52
12	37	2025-03-05 00:33:37	2025-08-31 02:50:07
13	40	2025-03-01 03:46:53	2025-09-07 22:03:19
14	41	2025-05-24 02:21:12	2025-08-18 19:17:43
15	42	2025-04-26 12:55:31	2025-08-09 05:15:50