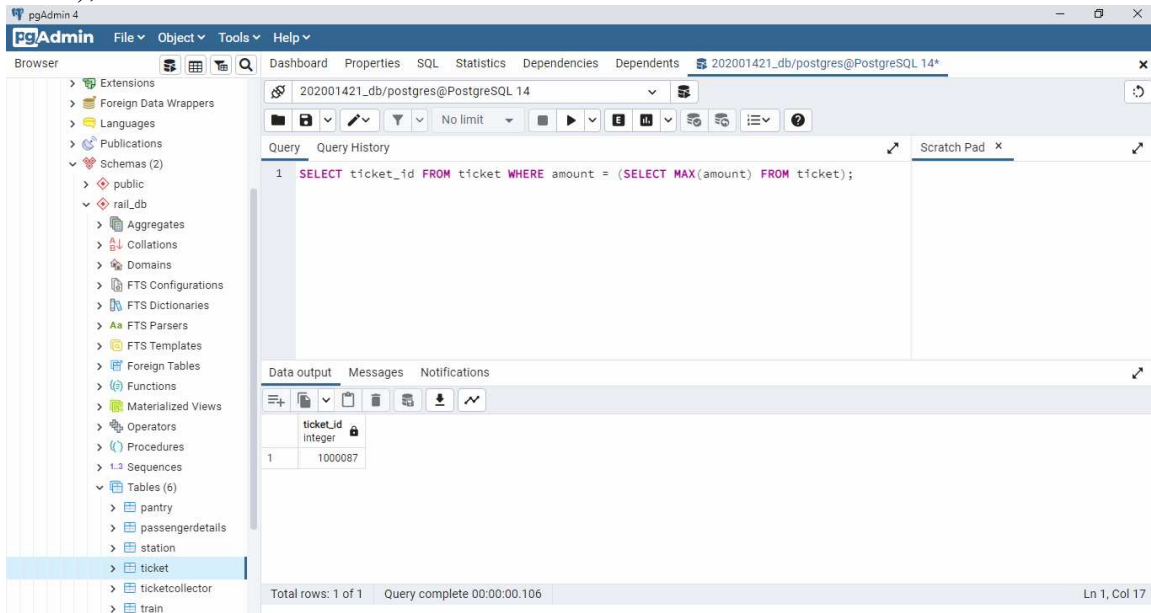


202001421 Lab 2

1: Display ticket id for maximum ticket amount.

SELECT ticket_id FROM ticket WHERE amount = (SELECT MAX(amount) FROM ticket);



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, with the 'ticket' table under the 'rail_db' schema selected. The main query editor contains the SQL query: `SELECT ticket_id FROM ticket WHERE amount = (SELECT MAX(amount) FROM ticket);`. The 'Data output' tab at the bottom shows the result of the query, which is a single row with the ticket_id 1000087.

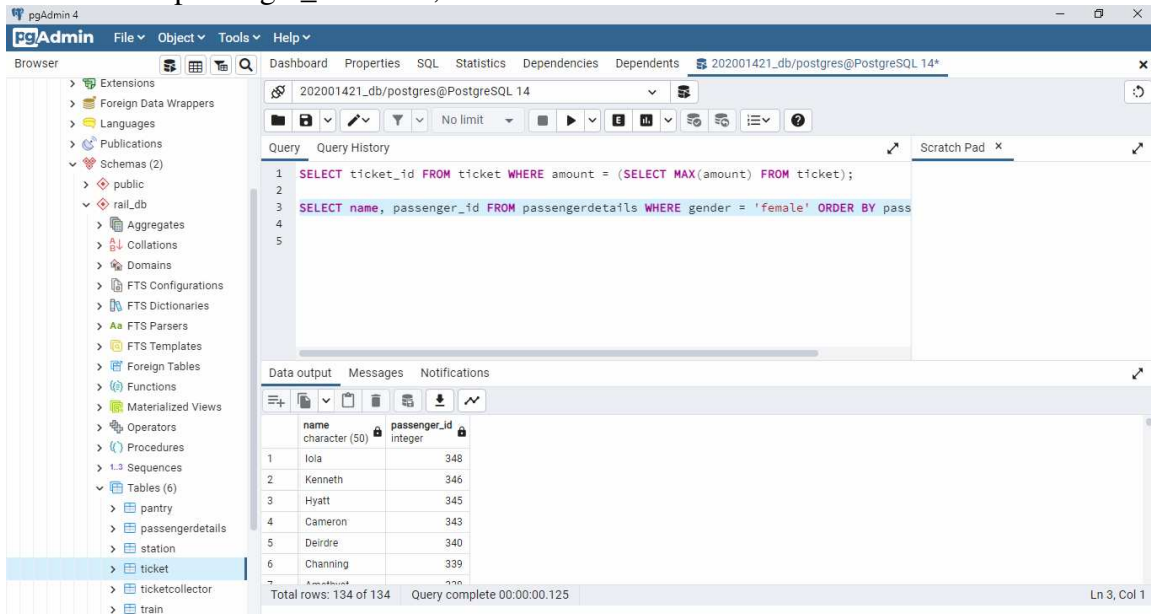
ticket_id
1000087

Total rows: 1 of 1 Query complete 00:00:00.106 Ln 1, Col 17

1 row

2: Print the Name of female passengers in descending order of passenger_id.

SELECT name, passenger_id FROM passengerdetails WHERE gender = 'female' ORDER BY passenger_id DESC;



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, with the 'passengerdetails' table under the 'rail_db' schema selected. The main query editor contains the SQL query: `SELECT name, passenger_id FROM passengerdetails WHERE gender = 'female' ORDER BY passenger_id DESC;`. The 'Data output' tab at the bottom shows the result of the query, which is a list of 6 female passengers ordered by passenger_id in descending order.

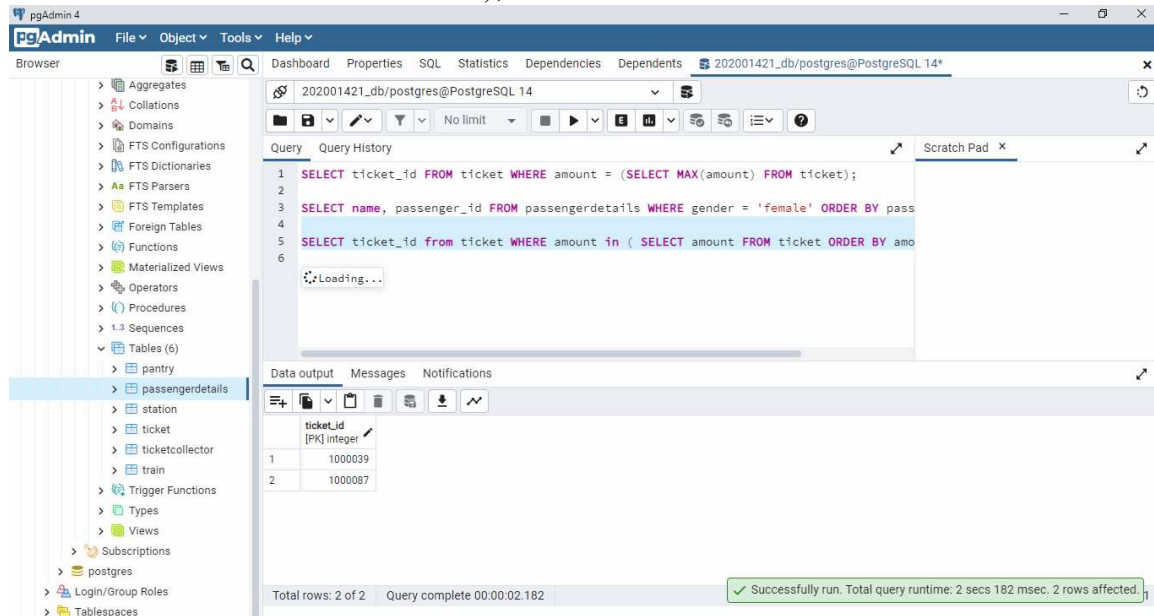
	name	passenger_id
1	Iola	348
2	Kenneth	346
3	Hyatt	345
4	Cameron	343
5	Deirdre	340
6	Channing	339

Total rows: 134 of 134 Query complete 00:00:00.125 Ln 3, Col 1

134 rows

3: Print ticket id of the highest 2 amounts of tickets.

SELECT ticket_id from ticket WHERE amount in (SELECT amount FROM ticket ORDER BY amount DESC LIMIT 2);



The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects, with 'passengerdetails' selected under the 'Tables (6)' category. The main pane shows a SQL query editor with the following query:

```
1 SELECT ticket_id FROM ticket WHERE amount = (SELECT MAX(amount) FROM ticket);
2
3 SELECT name, passenger_id FROM passengerdetails WHERE gender = 'female' ORDER BY pass
4
5 SELECT ticket_id from ticket WHERE amount in ( SELECT amount FROM ticket ORDER BY amo
6
```

The 'Data output' tab shows the results of the query:

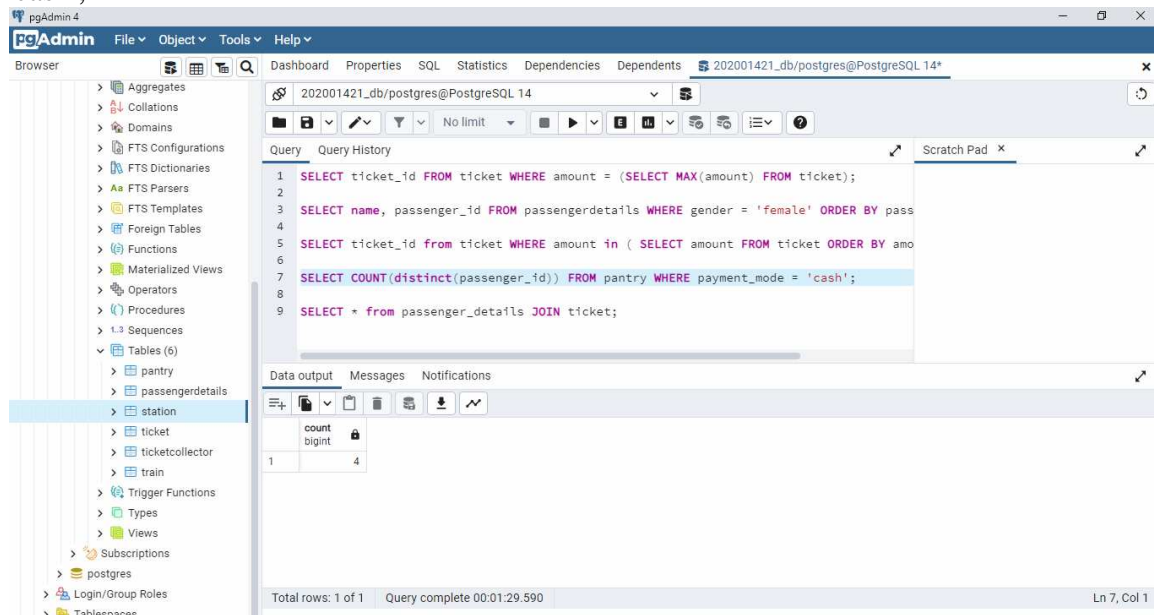
ticket_id [PK] integer
1000039
1000087

The status bar at the bottom indicates: 'Total rows: 2 of 2 Query complete 00:00:02.182 Successfully run. Total query runtime: 2 secs 182 msec. 2 rows affected.'

2 ROWS

4: Print count of passengers who did the payment of food in cash.

SELECT COUNT(distinct(passenger_id)) FROM pantry WHERE payment_mode = 'cash';



The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects, with 'pantry' selected under the 'Tables (6)' category. The main pane shows a SQL query editor with the following query:

```
1 SELECT ticket_id FROM ticket WHERE amount = (SELECT MAX(amount) FROM ticket);
2
3 SELECT name, passenger_id FROM passengerdetails WHERE gender = 'female' ORDER BY pass
4
5 SELECT ticket_id from ticket WHERE amount in ( SELECT amount FROM ticket ORDER BY amo
6
7 SELECT COUNT(distinct(passenger_id)) FROM pantry WHERE payment_mode = 'cash';
8
9 SELECT * from passenger_details JOIN ticket;
```

The 'Data output' tab shows the results of the query:

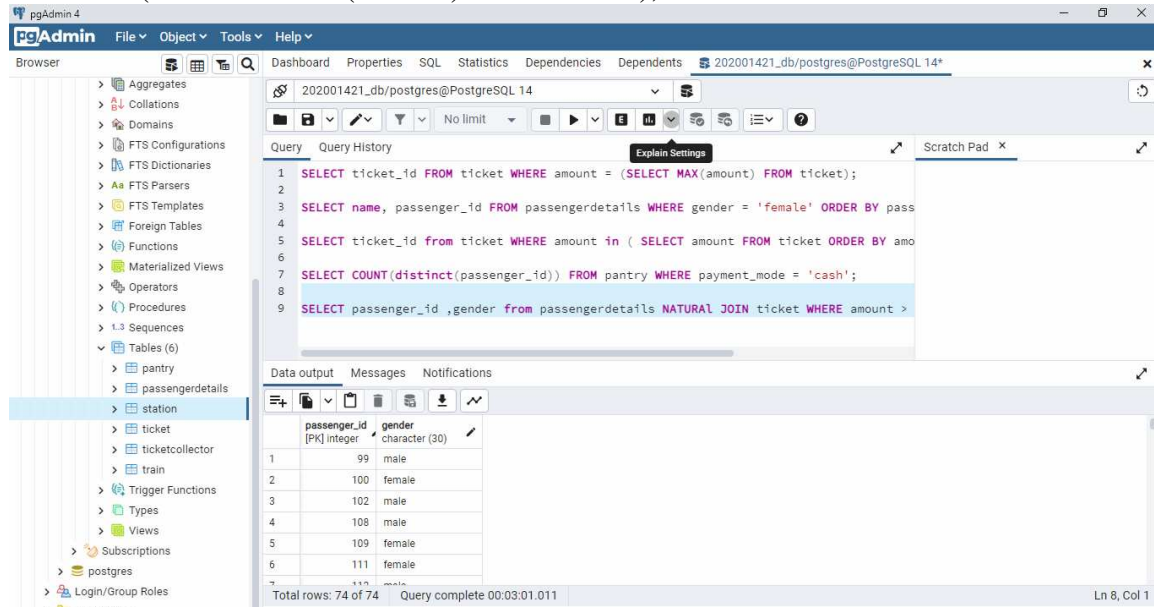
count bigint
4

The status bar at the bottom indicates: 'Total rows: 1 of 1 Query complete 00:01:29.590 Ln 7, Col 1'

1 row

5: Print Passenger id and gender of the passengers whose ticket amount is greater than avg amount value.

SELECT passenger_id ,gender from passengerdetails NATURAL JOIN ticket WHERE amount > (SELECT AVG(amount) FROM ticket);



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, with 'passengerdetails' and 'ticket' tables highlighted. The main window shows a SQL query in the 'Query' tab:

```
1 SELECT ticket_id FROM ticket WHERE amount = (SELECT MAX(amount) FROM ticket);
2
3 SELECT name, passenger_id FROM passengerdetails WHERE gender = 'female' ORDER BY pass
4
5 SELECT ticket_id from ticket WHERE amount in ( SELECT amount FROM ticket ORDER BY amo
6
7 SELECT COUNT(distinct(passenger_id)) FROM pantry WHERE payment_mode = 'cash';
8
9 SELECT passenger_id ,gender from passengerdetails NATURAL JOIN ticket WHERE amount >
```

The 'Data output' tab shows the results of the query:

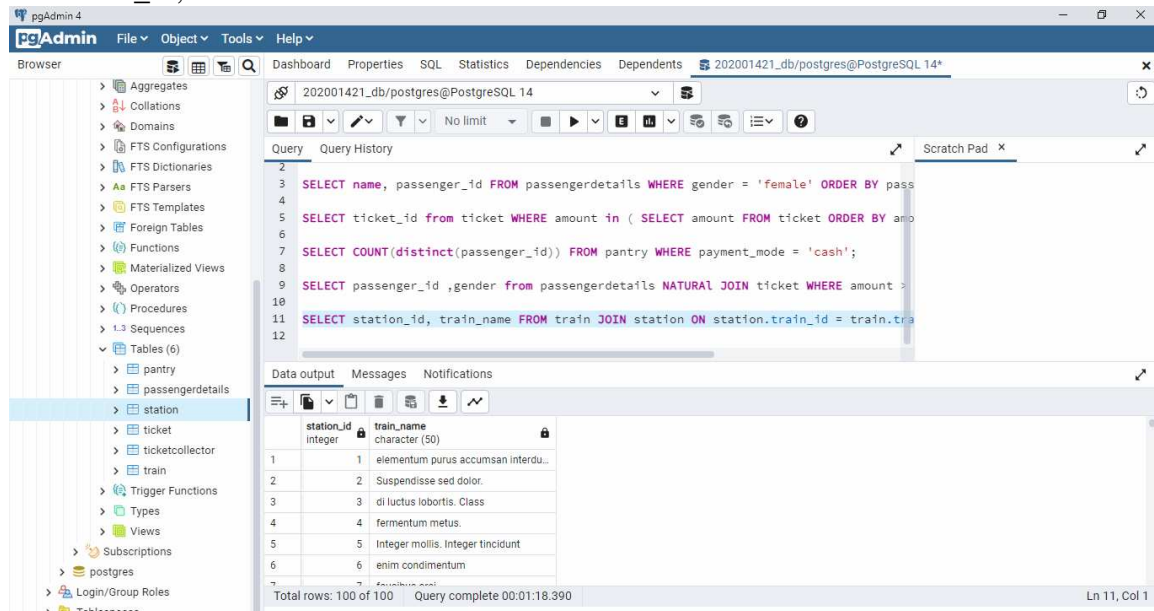
passenger_id [PK] integer	gender character (30)
1	99 male
2	100 female
3	102 male
4	108 male
5	109 female
6	111 female
7	112 male

Total rows: 74 of 74 Query complete 00:03:01.011 Ln 8, Col 1

74 Rows

6: Find the station id and the train name arriving at that station.

SELECT station_id, train_name FROM train JOIN station ON station.train_id = train.train_id;



The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, with 'station' and 'train' tables highlighted. The main window shows a SQL query in the 'Query' tab:

```
2
3 SELECT name, passenger_id FROM passengerdetails WHERE gender = 'female' ORDER BY pass
4
5 SELECT ticket_id from ticket WHERE amount in ( SELECT amount FROM ticket ORDER BY amo
6
7 SELECT COUNT(distinct(passenger_id)) FROM pantry WHERE payment_mode = 'cash';
8
9 SELECT passenger_id ,gender from passengerdetails NATURAL JOIN ticket WHERE amount >
10
11 SELECT station_id, train_name FROM train JOIN station ON station.train_id = train.train_id;
12
```

The 'Data output' tab shows the results of the query:

station_id integer	train_name character (50)
1	elementum purus accumsan interd...
2	Suspendisse sed dolor.
3	di luctus lobortis. Class
4	fermentum metus.
5	Integer mollis. Integer tincidunt
6	enim condimentum
7	...

Total rows: 100 of 100 Query complete 00:01:18.390 Ln 11, Col 1

100 rows

7: Print the count of different types of transactions

SELECT COUNT(payment_mode), payment_mode FROM pantry GROUP BY payment_mode;

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with 'Tables (6)' expanded and 'pantry' selected. The main pane shows a SQL query in the 'Query' tab:

```
3 SELECT name, passenger_id FROM passengerdetails WHERE gender = 'female' ORDER BY pass
4
5 SELECT ticket_id from ticket WHERE amount in ( SELECT amount FROM ticket ORDER BY amo
6
7 SELECT COUNT(distinct(passenger_id)) FROM pantry WHERE payment_mode = 'cash';
8
9 SELECT passenger_id ,gender from passengerdetails NATURAL JOIN ticket WHERE amount >
10
11 SELECT station_id, train_name FROM train JOIN station ON station.train_id = train.train_id
12
13 SELECT COUNT(payment_mode), payment_mode FROM pantry GROUP BY payment_mode;
```

The 'Data output' tab shows the results of the last query:

count	payment_mode
4	cash
6	online

Total rows: 2 of 2 Query complete 00:02:19.516 Ln 12, Col 1

2 ROWS

8: Find the id of ticket collectors whose age is less than avg age.

SELECT DISTINCT(tc_id) FROM ticketcollector WHERE age < (SELECT AVG(age) FROM ticketcollector);

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of the database structure, with 'Tables (6)' expanded and 'ticketcollector' selected. The main pane shows a SQL query in the 'Query' tab:

```
6
7 SELECT COUNT(distinct(passenger_id)) FROM pantry WHERE payment_mode = 'cash';
8
9 SELECT passenger_id ,gender from passengerdetails NATURAL JOIN ticket WHERE amount >
10
11 SELECT station_id, train_name FROM train JOIN station ON station.train_id = train.train_id
12
13 SELECT COUNT(payment_mode), payment_mode FROM pantry GROUP BY payment_mode;
14
15 SELECT DISTINCT(tc_id ) FROM ticketcollector WHERE age < (SELECT AVG(age) FROM ticketcollector);
16
```

The 'Data output' tab shows the results of the last query:

tc_id
3
23
24
8
10
9

Total rows: 13 of 13 Query complete 00:00:01.900 Ln 15, Col 1

13 rows

9: Find the names of trains which are taking a halt at a station and sort in order of station id.

```
SELECT train_name FROM train JOIN station ON station.train_id = train.train_id  
WHERE halt = 'Yes' ORDER BY station.station_id;
```

The screenshot shows the pgAdmin 4 interface. The left sidebar displays a tree view of database objects, with 'ticketcollector' selected under the 'Tables (6)' category. The main pane shows a SQL query editor with the following code:

```
8  
9 SELECT passenger_id ,gender from passengerdetails NATURAL JOIN ticket WHERE amount >  
10  
11 SELECT station_id, train_name FROM train JOIN station ON station.train_id = train.train_id  
12  
13 SELECT COUNT(payment_mode), payment_mode FROM pantry GROUP BY payment_mode;  
14  
15 SELECT DISTINCT(tc_id ) FROM ticketcollector WHERE age < (SELECT AVG(age) FROM ticket  
16  
17 SELECT train_name FROM train JOIN station ON station.train_id = train.train_id WHERE  
18 CREATE VIEW mx as SELECT MAX(amount) from ticket;
```

The 'Data output' tab is active, displaying the results of the query. The first result is a table with 56 rows and 1 column, 'train_name', which is a character type of length 50. The data is as follows:

train_name
di luctus lobortis. Class
fermentum metus.
Integer mollis. Integer tincidunt
enim condimentum
dui. Cras pellentesque. Sed dictum.
ultrices sem magna nec

The status bar at the bottom indicates 'Total rows: 56 of 56' and 'Query complete 00:00:00.838'.

56 rows

10: Create a view of the maximum amount.

```
CREATE OR REPLACE VIEW mx as SELECT MAX(amount) from ticket;
```

The screenshot shows the pgAdmin 4 interface. The left sidebar is the same as in the previous screenshot. The main pane shows a SQL query editor with the following code:

```
9 SELECT passenger_id ,gender from passengerdetails NATURAL JOIN ticket WHERE amount >  
10  
11 SELECT station_id, train_name FROM train JOIN station ON station.train_id = train.train_id  
12  
13 SELECT COUNT(payment_mode), payment_mode FROM pantry GROUP BY payment_mode;  
14  
15 SELECT DISTINCT(tc_id ) FROM ticketcollector WHERE age < (SELECT AVG(age) FROM ticket  
16  
17 SELECT train_name FROM train JOIN station ON station.train_id = train.train_id WHERE  
18  
19 CREATE OR REPLACE VIEW mx as SELECT MAX(amount) from ticket;
```

The 'Messages' tab is active, displaying the message: 'CREATE VIEW' and 'Query returned successfully in 130 msec.' The status bar at the bottom indicates 'Total rows: 56 of 56' and 'Query complete 00:00:00.130'.

11: Create a view for station details then insert and display some new station ids to the recently created view.

```
CREATE OR REPLACE VIEW station_details as (SELECT * from station);  
INSERT INTO station_details VALUES('101','Z5T 3C4','08:45:00','106','Yes');  
SELECT * FROM station_details WHERE station_id = 101;
```

The first screenshot shows the pgAdmin 4 interface with a SQL query window. The query is:

```
12 SELECT COUNT(payment_mode), payment_mode FROM pantry GROUP BY payment_mode;  
13  
14 SELECT DISTINCT(tc_id ) FROM ticketcollector WHERE age < (SELECT AVG(age) FROM ticket  
15  
16 SELECT train_name FROM train JOIN station ON station.train_id = train.train_id WHERE  
17  
18 CREATE OR REPLACE VIEW mx as SELECT MAX(amount) from ticket;  
19  
20  
21 CREATE OR REPLACE VIEW station_details as (SELECT * from station);  
22 INSERT INTO station_details VALUES('101','Z5T 3C4','08:45:00','106','Yes');
```

The Data output tab shows the message: "INSERT 0 1" and "Query returned successfully in 843 msec." The status bar indicates: "Total rows: 100 of 100 Query complete 00:00:00.843 Successfully run. Total query runtime: 2 min 57 secs. 100 rows affected."

The second screenshot shows the same pgAdmin 4 interface with a different SQL query window. The query is:

```
18 SELECT train_name FROM train JOIN station ON station.train_id = train.train_id WHERE  
19  
20 CREATE OR REPLACE VIEW mx as SELECT MAX(amount) from ticket;  
21  
22 CREATE OR REPLACE VIEW station_details as (SELECT * from station);  
23 INSERT INTO station_details VALUES('101','Z5T 3C4','08:45:00','106','Yes');  
24 SELECT * FROM station_details WHERE station_id = 101;  
25  
26 CREATE OR REPLACE VIEW rate_details as (SELECT AVG(rate) from pantry);  
27 SELECT * FROM rate_details;  
28 INSERT INTO pantry VALUES('9999', 'food1', '100', 'Yes', 'Online', '10090');  
29 INSERT INTO pantry VALUES('9998', 'food2', '100', 'Yes', 'Online', '10091');
```

The Data output tab shows a table with the following data:

station_id	name	arrival_time	train_id	halt
101	Z5T 3C4	08:45:00	106	Yes

The status bar indicates: "Total rows: 1 of 1 Query complete 00:00:00.342 Ln 24, Col 1"

12: Create a view of the rate and put the average there. add 4 more food items to the pantry table and then display the contents of the view with the updated values.

```
CREATE OR REPLACE VIEW rate_details as (SELECT AVG(rate) from pantry);  
SELECT * FROM rate_details;
```

```

INSERT INTO pantry VALUES('9999', 'food1', '100', 'Yes', 'Online', '99');
INSERT INTO pantry VALUES('9998', 'food2', '100', 'Yes', 'Online', '102');
INSERT INTO pantry VALUES('9997', 'food3', '100', 'Yes', 'Online', '101');
INSERT INTO pantry VALUES('9996', 'food4', '100', 'Yes', 'Online', '100');
SELECT * FROM rate_details;

```

pgAdmin 4

202001421_db/postgres@PostgreSQL 14*

Query

```

23 INSERT INTO station_details VALUES('101','Z5T 3C4','08:45:00','100','Yes');
24 SELECT * FROM station_details WHERE station_id = 101;
25
26 CREATE OR REPLACE VIEW rate_details as (SELECT AVG(rate) from pantry);
27 SELECT * FROM rate_details;
28 INSERT INTO pantry VALUES('9999', 'food1', '100', 'Yes', 'Online', '99');
29 INSERT INTO pantry VALUES('9998', 'food2', '100', 'Yes', 'Online', '102');
30 INSERT INTO pantry VALUES('9997', 'food3', '100', 'Yes', 'Online', '101');
31 INSERT INTO pantry VALUES('9996', 'food4', '100', 'Yes', 'Online', '100');
32 SELECT * FROM rate_details;
33
34 CREATE OR REPLACE VIEW pass_food_detail as (SELECT * FROM passengerdetails JOIN pan

```

Data output

avg numeric
538.6000000

Total rows: 1 of 1 Query complete 00:00:00.088 Successfully run. Total query runtime: 88 msec. 1 rows affected.

pgAdmin 4

202001421_db/postgres@PostgreSQL 14*

Query

```

23 INSERT INTO station_details VALUES('101','Z5T 3C4','08:45:00','100','Yes');
24 SELECT * FROM station_details WHERE station_id = 101;
25
26 CREATE OR REPLACE VIEW rate_details as (SELECT AVG(rate) from pantry);
27 SELECT * FROM rate_details;
28 INSERT INTO pantry VALUES('9999', 'food1', '100', 'Yes', 'Online', '99');
29 INSERT INTO pantry VALUES('9998', 'food2', '100', 'Yes', 'Online', '102');
30 INSERT INTO pantry VALUES('9997', 'food3', '100', 'Yes', 'Online', '101');
31 INSERT INTO pantry VALUES('9996', 'food4', '100', 'Yes', 'Online', '100');
32 SELECT * FROM rate_details;
33
34 CREATE OR REPLACE VIEW pass_food_detail as (SELECT * FROM passengerdetails JOIN pan

```

Data output

INSERT 0 1

Query returned successfully in 102 msec.

Total rows: 1 of 1 Query complete 00:00:00.102 Query returned successfully in 102 msec.

pgAdmin 4

File Object Tools Help

Dashboard Properties SQL Statistics Dependencies Dependents 202001421_db/postgres@PostgreSQL 14*

Browser

- Aggregates
- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (6)
 - pantry
 - passengerdetails
 - station
 - ticket
 - ticketcollector
 - train
- Trigger Functions
- Types
- Views (3)
 - mx
 - rate_details
 - station_details
- Subscriptions

Query Query History

```
23 INSERT INTO station_details VALUES('101','ZST 3C4','08:45:00','100','Yes');
24 SELECT * FROM station_details WHERE station_id = 101;
25
26 CREATE OR REPLACE VIEW rate_details as (SELECT AVG(rate) from pantry);
27 SELECT * FROM rate_details;
28 INSERT INTO pantry VALUES('9999','food1','100','Yes','Online','99');
29 INSERT INTO pantry VALUES('9998','food2','100','Yes','Online','102');
30 INSERT INTO pantry VALUES('9997','food3','100','Yes','Online','101');
31 INSERT INTO pantry VALUES('9996','food4','100','Yes','Online','100');
32 SELECT * FROM rate_details;
33
34 CREATE OR REPLACE VIEW pass_food_details as (SELECT * FROM passengerdetails JOIN pan
```

Data output Messages Notifications

INSERT 0 1

Query returned successfully in 111 msec.

Total rows: 1 of 1 Query complete 00:00:00.111

✓ Query returned successfully in 111 msec.

pgAdmin 4

File Object Tools Help

Dashboard Properties SQL Statistics Dependencies Dependents 202001421_db/postgres@PostgreSQL 14*

Browser

- Aggregates
- Collations
- Domains
- FTS Configurations
- FTS Dictionaries
- FTS Parsers
- FTS Templates
- Foreign Tables
- Functions
- Materialized Views
- Operators
- Procedures
- Sequences
- Tables (6)
 - pantry
 - passengerdetails
 - station
 - ticket
 - ticketcollector
 - train
- Trigger Functions
- Types
- Views (3)
 - mx
 - rate_details
 - station_details
- Subscriptions

Query Query History

```
23 INSERT INTO station_details VALUES('101','ZST 3C4','08:45:00','100','Yes');
24 SELECT * FROM station_details WHERE station_id = 101;
25
26 CREATE OR REPLACE VIEW rate_details as (SELECT AVG(rate) from pantry);
27 SELECT * FROM rate_details;
28 INSERT INTO pantry VALUES('9999','food1','100','Yes','Online','99');
29 INSERT INTO pantry VALUES('9998','food2','100','Yes','Online','102');
30 INSERT INTO pantry VALUES('9997','food3','100','Yes','Online','101');
31 INSERT INTO pantry VALUES('9996','food4','100','Yes','Online','100');
32 SELECT * FROM rate_details;
33
34 CREATE OR REPLACE VIEW pass_food_details as (SELECT * FROM passengerdetails JOIN pan
```

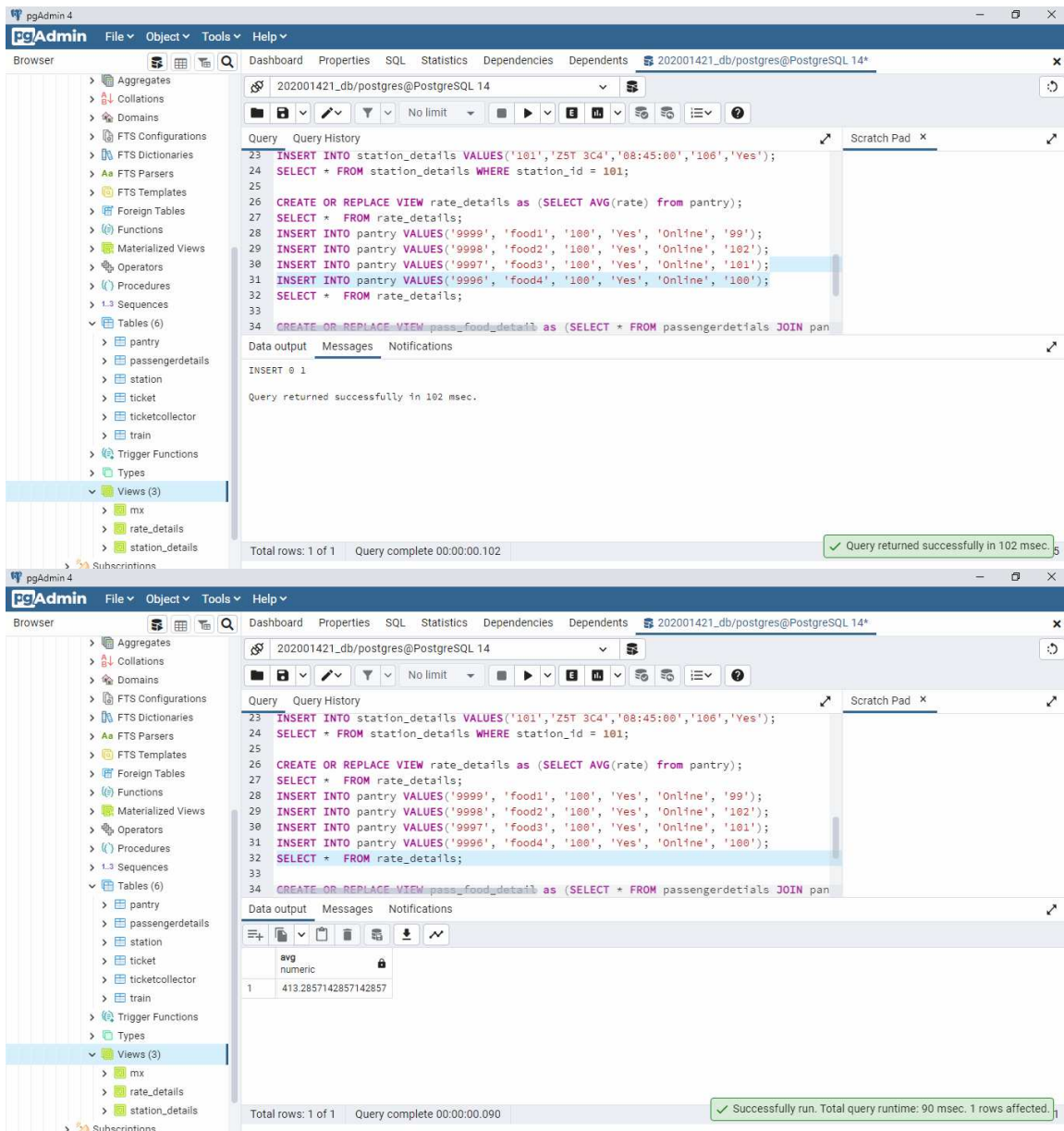
Data output Messages Notifications

INSERT 0 1

Query returned successfully in 89 msec.

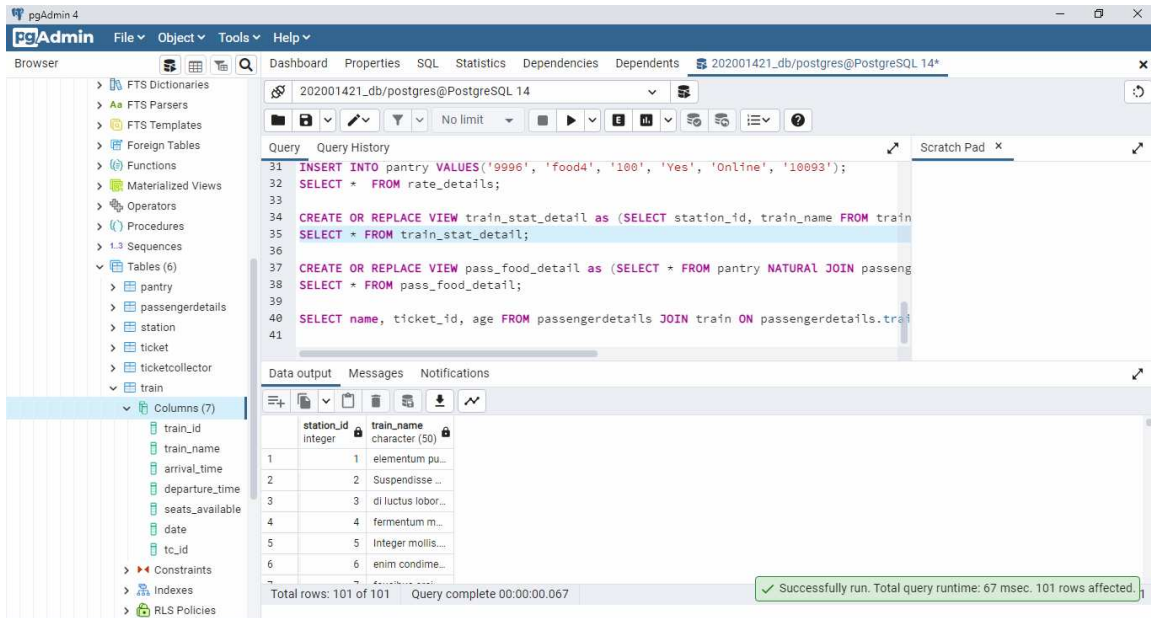
Total rows: 1 of 1 Query complete 00:00:00.089

✓ Query returned successfully in 89 msec.



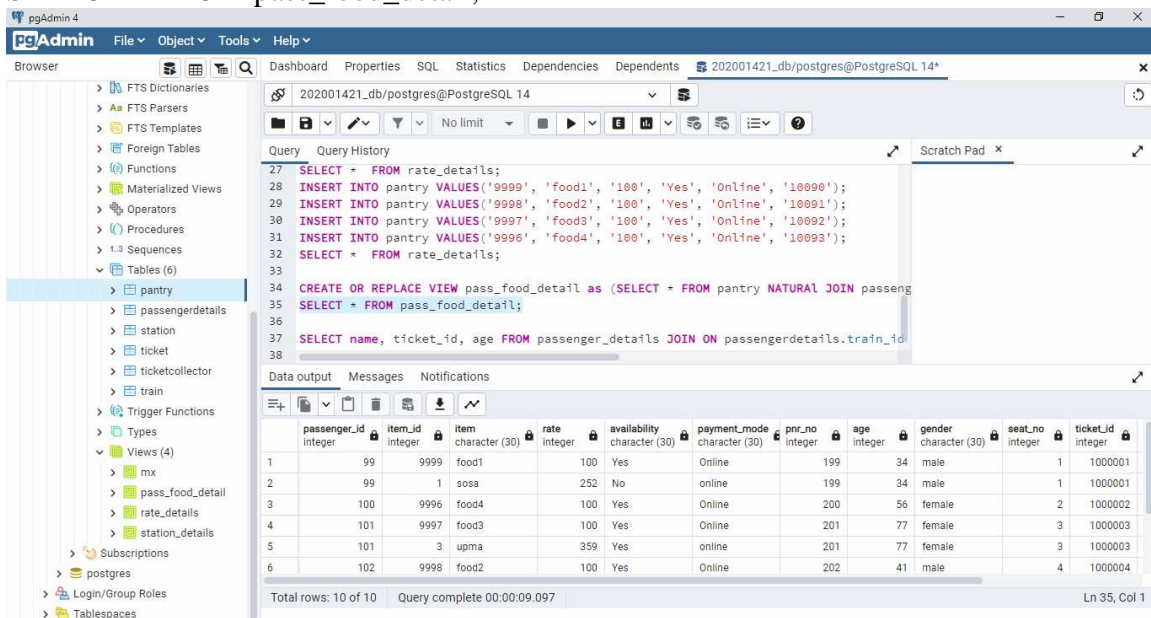
13: Create a view with all the train names and the station_id on which it is arriving in ascending order of station id.

```
CREATE OR REPLACE VIEW train_stat_detail as (SELECT station_id, train_name
FROM train JOIN station ON station.train_id = train.train_id);
SELECT * FROM train_stat_detail;
```



14: Display all the details of passengers whose payment mode was online for food items and create a view for the same.

CREATE OR REPLACE VIEW pass_food_detail as (SELECT * FROM pantry NATURAL JOIN passengerdetails WHERE payment_mode = 'Online' or payment_mode = 'online');
SELECT * FROM pass_food_detail;



15: Display the name, ticket id and age of the passengers who are travelling in a train with the number of seats available is greater than 10 in decreasing order of the ticket id.

SELECT name, ticket_id, age FROM passengerdetails JOIN train ON
passengerdetails.train_id = train.train_id WHERE train.seats_available > 10 ORDER BY
ticket_id DESC;

The screenshot shows the pgAdmin 4 interface. The left sidebar displays the database structure, with the 'passengerdetails' table selected under the 'Tables (6)' category. The main window shows a SQL query in the 'Query' tab, which is a multi-part query including INSERT statements, a CREATE VIEW statement, and a SELECT statement. The 'Data output' tab shows the results of the SELECT statement, which is a table with 6 rows and 3 columns: name, ticket_id, and age. The status bar at the bottom indicates 'Total rows: 127 of 127' and 'Query complete 00:00:00.076'.

Query:

```

28 INSERT INTO pantry VALUES('9999', 'food1', '100', 'Yes', 'Online', '10090');
29 INSERT INTO pantry VALUES('9998', 'food2', '100', 'Yes', 'Online', '10091');
30 INSERT INTO pantry VALUES('9997', 'food3', '100', 'Yes', 'Online', '10092');
31 INSERT INTO pantry VALUES('9996', 'food4', '100', 'Yes', 'Online', '10093');
32 SELECT * FROM rate_details;
33
34 CREATE OR REPLACE VIEW pass_food_detail as (SELECT * FROM pantry NATURAL JOIN passeng
35 SELECT * FROM pass_food_detail;
36
37 SELECT name, ticket_id, age FROM passengerdetails JOIN train ON passengerdetails.train
38

```

Data output:

	name character (50)	ticket_id integer	age integer
1	Iola	1000250	38
2	Guinevere	1000249	66
3	Cooper	1000243	60
4	Channing	1000241	45
5	Micah	1000239	37
6	Thane	1000237	37

Total rows: 127 of 127 Query complete 00:00:00.076 Ln 37, Col 1