

RunSQLiteSQLiteSQLiteSQLiteSQLiteSQLite

83

84 INSERT INTO baggage VALUES

85 (1, 1, 28.5, '2023-11-10', '2023-11-05'),

86 (2, 2, 18.0, '2023-12-01', '2023-12-01'),

87 (3, 3, 30.0, '2024-01-15', '2024-01-10'),

88 (4, 4, 26.5, '2023-10-20', '2023-10-18');

89

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

91 SELECT

92 first_name || ' ' || last_name AS full_name,

93 strftime('%m/%d/%Y', birth_date) AS formatted_birth_date

94 FROM passengers;

95

full_name	formatted_birth_date
Ivan Ivan	03/21/1995
Aigerim Nurlanova	07/11/1999
Sultan Bekzada	01/15/2003
Aliya Aliya	05/10/1992

History

SyntaxHistory

SQLite

DROP TABLE IF EXISTS passengers;

DROP TABLE IF EXISTS tickets;

DROP TABLE IF EXISTS flights;

DROP TA

...

11:18:37

SQLite

DROP TABLE IF EXISTS passengers;

DROP TABLE IF EXISTS tickets;

DROP TABLE IF EXISTS flights;

DROP TA

...

Help: SQLITE_ERROR: sqlite3 result code 1:

no such function: DATE_FORMAT

You have reached the hint limit in your SQL

code according to your plan.

```
Run SQLite SQLite SQLite SQLite SQLite SQLite SQLite
83
84 INSERT INTO baggage VALUES
85 (1, 1, 28.5, '2023-11-10', '2023-11-05'),
86 (2, 2, 18.0, '2023-12-01', '2023-12-01'),
87 (3, 3, 30.0, '2024-01-15', '2024-01-10'),
88 (4, 4, 26.5, '2023-10-20', '2023-10-18');
89
90 -- 4. Retrieve airports that contain the word "Reginal" and "Air" in their names.
91 SELECT airline_name
92 FROM airlines
93 WHERE airline_name LIKE '%Reginal%'
94 AND airline_name LIKE '%Air%';
95
```


```
airline_name
```

History

Syntax

History

 SQLite



```
DROP TABLE IF EXISTS passengers;
DROP TABLE IF EXISTS tickets;
DROP TABLE IF EXISTS flights;
DROP TA
...
```

11:16:39

 SQLite



```
DROP TABLE IF EXISTS passengers;
DROP TABLE IF EXISTS tickets;
DROP TABLE IF EXISTS flights;
DROP TA
...
```

11:15:02

your

84 INSERT INTO baggage VALUES

85 (1, 1, 28.5, '2023-11-10', '2023-11-05'),

86 (2, 2, 18.0, '2023-12-01', '2023-12-01'),

87 (3, 3, 30.0, '2024-01-15', '2024-01-10'),

88 (4, 4, 26.5, '2023-10-20', '2023-10-18');

89

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

91 SELECT flight_id

92 FROM flights

93 WHERE airline_id IN (1, 2)

94 GROUP BY flight_id

95 HAVING COUNT(DISTINCT airline_id) = 2;

96

! flight_id

SyntaxHistory

SQLite

▶

🔗

⌵

⋮

DROP TABLE IF EXISTS passengers;

DROP TABLE IF EXISTS tickets;

DROP TABLE IF EXISTS flights;

DROP TA

...

11:15:02

SQLite

▶

🔗

⌵

⋮

DROP TABLE IF EXISTS passengers;

DROP TABLE IF EXISTS tickets;

DROP TABLE IF EXISTS flights;

DROP TA

...

11:13:39

SQLite

▶

🔗

⌵

⋮

DROP TABLE IF EXISTS passengers;

Run

SQLite

SQLite

SQLite

SQLite

SQLite

SQLite

81 (280, 'Website', 9500, '2023-11-20', 'Pending'),

82 (301, 'MobileApp', 15000, '2024-03-01', 'Checked');

83

84 INSERT INTO baggage VALUES

85 (1, 1, 28.5, '2023-11-10', '2023-11-05'),

86 (2, 2, 18.0, '2023-12-01', '2023-12-01'),

87 (3, 3, 30.0, '2024-01-15', '2024-01-10'),

88 (4, 4, 26.5, '2023-10-20', '2023-10-18');

89

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

91 SELECT REPLACE(airline_name, 'Air', 'Aero') AS modified_airline_name

92 FROM airlines;

93

modified_airline_name

Aero China

airlines - TABLE

baggage - TABLE

bookings - TABLE

demo - TABLE

flights - TABLE

passengers - TABLE

tickets - TABLE

History

SyntaxHistory

SQLite

DROP TABLE IF EXISTS passengers;

DROP TABLE IF EXISTS tickets;

DROP TABLE IF EXISTS flights;

DROP TA

11:13:39

XISTS passengers;

XISTS tickets;

```
airlines - TABLE
baggage - TABLE
bookings - TABLE
demo - TABLE
flights - TABLE
passengers - TABLE
tickets - TABLE

└─ airline_id AIRLINES:int, FLIGHTS:int
└─ airline_name AIRLINES:varchar(100)
```

Syntax History

11:13

11.12

RunSQLiteSQLiteSQLiteSQLiteSQLiteSQLite

102

103

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

105 SELECT flight_id, destination, airline_id

106 FROM flights

107 WHERE actual_arrival > scheduled_arrival;

108

109

110

111

112

113

114

flight_id	destination	airline_id
1	China	1
3	Kazakhstan	3

History

SyntaxHistory

SQLite

DROP TABLE IF EXISTS passengers;

DROP TABLE IF EXISTS tickets;

DROP TABLE IF EXISTS flights;

DROP TA

...

11:27:48

SQLite

DROP TABLE IF EXISTS passengers;

DROP TABLE IF EXISTS tickets;

DROP TABLE IF EXISTS flights;

DROP TA

...

11:26:59

SQLite

DROP TABLE IF EXISTS passengers;

DROP TABLE IF EXISTS tickets;

DROP TABLE IF EXISTS flights;

DROP TA

...

11:26:59

Run SQLite SQLite SQLite SQLite SQLite SQLite SQLite

```
102
103
104 -- 9. Find number of airline names in each airline country.
105 SELECT
106     country,
107     COUNT(*) AS airline_count
108 FROM airlines
109 GROUP BY country;
110
111
112
113
114
```

History

Syntax History

```
SQLite
DROP TABLE IF EXISTS passengers;
DROP TABLE IF EXISTS tickets;
DROP TABLE IF EXISTS flights;
DROP TA
...
11:26:59
```

```
SQLite
DROP TABLE IF EXISTS passengers;
DROP TABLE IF EXISTS tickets;
DROP TABLE IF EXISTS flights;
DROP TA
...
```

```
SQLite
DROP TABLE IF EXISTS passengers;
DROP TABLE IF EXISTS tickets;
DROP TABLE IF EXISTS flights;
DROP TA
```

```
102
103
104 -- 8. Create a query that categorizes ticket prices based on their price as "Cheap," "Medium" or "Expensive.".
105 SELECT
106     ticket_id,
107     price,
108     CASE
109         WHEN price < 16000 THEN 'Cheap'
110         WHEN price BETWEEN 16000 AND 20000 THEN 'Medium'
111         ELSE 'Expensive'
112     END AS price_category
113 FROM tickets;
114
```

ticket_id	price	price_category
1	15000	Cheap
2	20000	Medium
3	18000	Medium
4	25000	Expensive

Syntax History

SQLite

```
DROP TABLE IF EXISTS passengers;
DROP TABLE IF EXISTS tickets;
DROP TABLE IF EXISTS flights;
DROP TA
...
```

11:26:30

SQLite

```
DROP TABLE IF EXISTS passengers;
DROP TABLE IF EXISTS tickets;
DROP TABLE IF EXISTS flights;
DROP TA
...
```

11:24:51

SQLite

```
DROP TABLE IF EXISTS passengers;
DROP TABLE IF EXISTS tickets;
```



```
99 UPDATE flights
100 SET scheduled_arrival = '2024-03-12 18:00', actual_arrival = '2024-03-12 18:40'
101 WHERE flight_id = 3;
102
103
104 -- 6. Find flight numbers that have been delayed based on the actual arrival time.
105 SELECT flight_id, destination, scheduled_arrival, actual_arrival
106 FROM flights
107 WHERE actual_arrival > scheduled_arrival;
108
109
110
111
```

flight_id	destination	scheduled_arrival	actual_arrival
1	China	2024-03-10 10:00	2024-03-10 10:30
3	Kazakhstan	2024-03-12 18:00	2024-03-12 18:40

DROP TABLE IF EXISTS passengers;
DROP TABLE IF EXISTS tickets;
DROP TABLE IF EXISTS flights;
DROP TA
...

Help: SQLITE_ERROR: sqlite3 result code 1
no such column: YEAR

You have reached the hint limit in your SQL code according to your plan.

AI Help

11:21:45

SQLite

DROP TABLE IF EXISTS passengers;
DROP TABLE IF EXISTS tickets;
DROP TABLE IF EXISTS flights;
DROP TA
...

```

87 (4, 4, 26.5, '2023-10-20', '2023-10-18');
88 (4, 4, 26.5, '2023-10-20', '2023-10-18');
89
90 -- 7.
91 SELECT
92     first_name || ' ' || last_name AS full_name,
93     CASE
94         WHEN (CAST(strftime('%Y', 'now') AS INTEGER) - CAST(strftime('%Y', birth_date) AS INTEGER)) BETWEEN 18 AND 35 THEN 'Young'
95         WHEN (CAST(strftime('%Y', 'now') AS INTEGER) - CAST(strftime('%Y', birth_date) AS INTEGER)) BETWEEN 36 AND 55 THEN 'Middle-aged'
96         ELSE 'Other'
97     END AS age_group
98 FROM passengers;
99
100

```

full_name	age_group
Ivan Ivan	Young
Algerim Nurlanova	Young
Sultan Bekzada	Young
Aliya Aliya	Young

Syntax History

SQLite

```

DROP TABLE IF EXISTS passengers;
DROP TABLE IF EXISTS tickets;
DROP TABLE IF EXISTS flights;
DROP TA
...

```

11:23:22

SQLite

```

DROP TABLE IF EXISTS passengers;
DROP TABLE IF EXISTS tickets;
DROP TABLE IF EXISTS flights;
DROP TA
...

```

Help: SQLITE_ERROR: sqlite3 result code 1:
no such column: YEAR

You have reached the hint limit in your SQL code according to your plan.