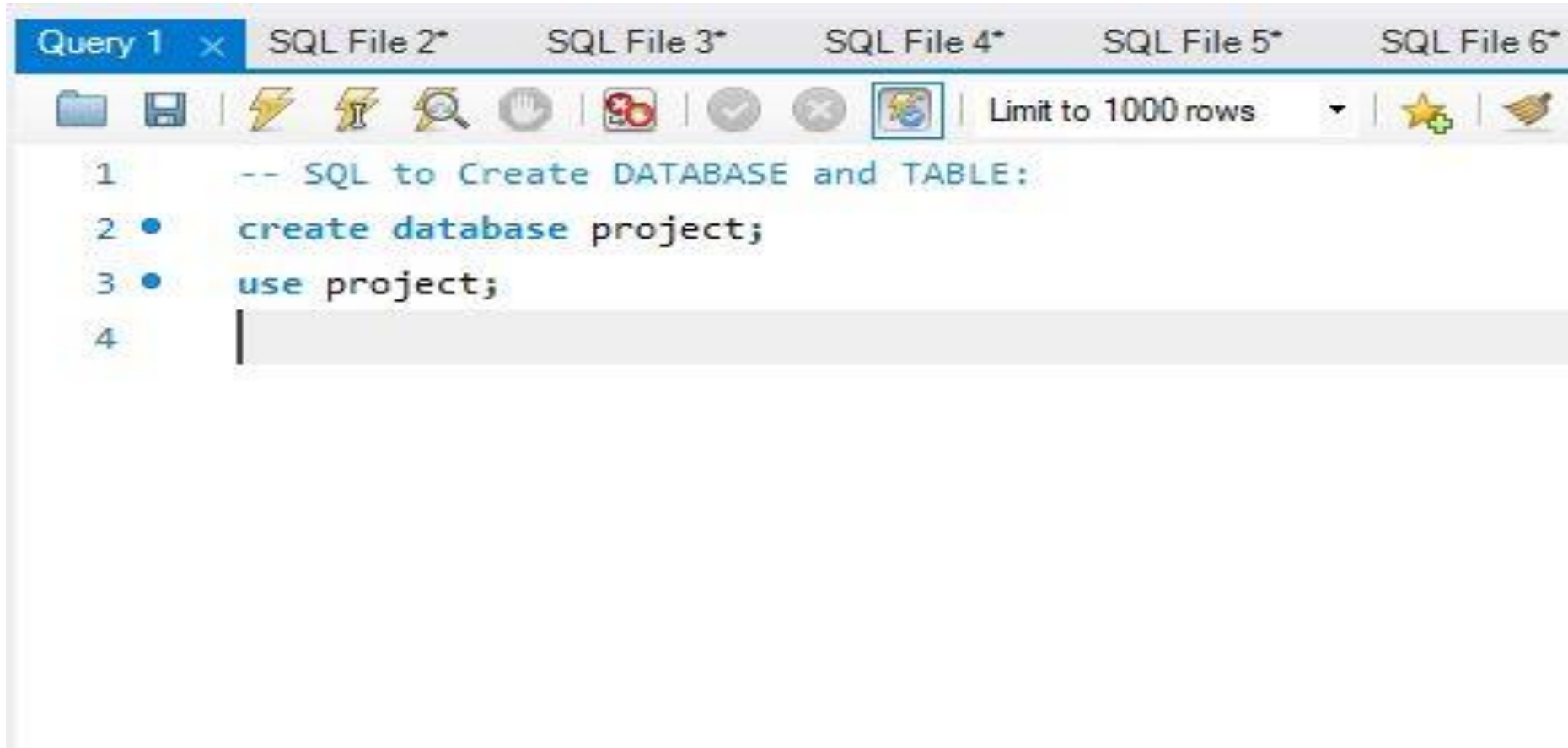


SQL Based -Customer Order Analysis


-- SQL to Create DATABASE and TABLE:

A screenshot of a SQL IDE interface. The top bar shows several tabs: 'Query 1' (active), 'SQL File 2*', 'SQL File 3*', 'SQL File 4*', 'SQL File 5*', and 'SQL File 6*'. Below the tabs is a toolbar with various icons for file operations, execution, and settings. A dropdown menu is open, showing 'Limit to 1000 rows'. The main editor area displays a SQL query with line numbers 1 through 4 on the left. The query text is: 1 -- SQL to Create DATABASE and TABLE: 2 create database project; 3 use project; 4 |

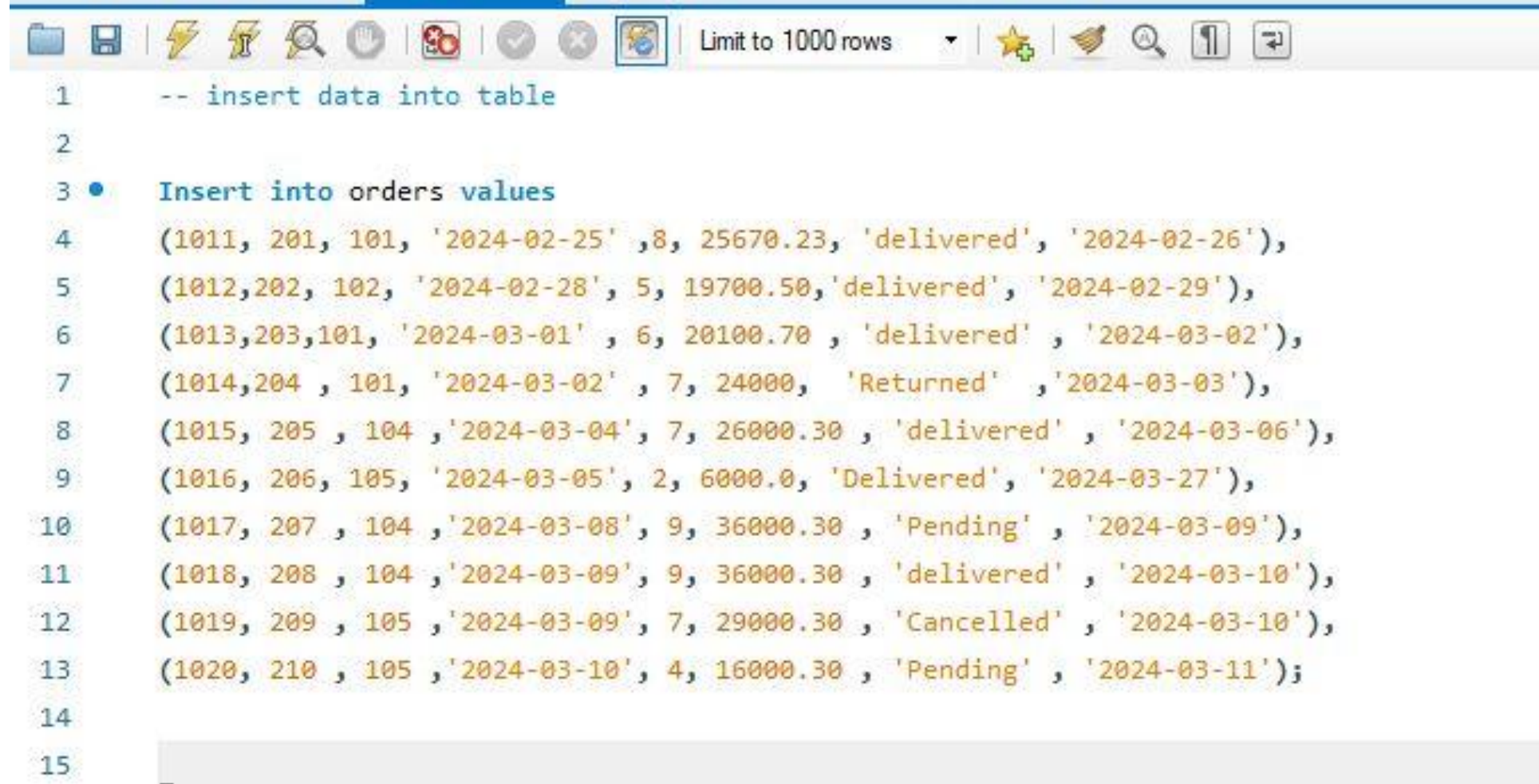
```
1 -- SQL to Create DATABASE and TABLE:
2 create database project;
3 use project;
4 |
```

-- Create table

```
-- Create table
```

-  Create table orders (
order_id int primary key,
customer_id int,
product_id int,
order_date date,
quantity int,
total_amount decimal(10,2),
order_status varchar (20),
shipping_date date
);

-- insert data into table



The screenshot shows a SQL IDE window with a toolbar at the top. The toolbar includes icons for file operations (folder, save), execution (lightning bolt, play), search (magnifying glass), and other standard IDE functions. A dropdown menu is set to "Limit to 1000 rows". The main text area contains an SQL insert statement with 15 line numbers on the left. The statement is as follows:

```
1  -- insert data into table
2
3  • Insert into orders values
4  (1011, 201, 101, '2024-02-25', 8, 25670.23, 'delivered', '2024-02-26'),
5  (1012, 202, 102, '2024-02-28', 5, 19700.50, 'delivered', '2024-02-29'),
6  (1013, 203, 101, '2024-03-01', 6, 20100.70, 'delivered', '2024-03-02'),
7  (1014, 204, 101, '2024-03-02', 7, 24000, 'Returned', '2024-03-03'),
8  (1015, 205, 104, '2024-03-04', 7, 26000.30, 'delivered', '2024-03-06'),
9  (1016, 206, 105, '2024-03-05', 2, 6000.0, 'Delivered', '2024-03-27'),
10 (1017, 207, 104, '2024-03-08', 9, 36000.30, 'Pending', '2024-03-09'),
11 (1018, 208, 104, '2024-03-09', 9, 36000.30, 'delivered', '2024-03-10'),
12 (1019, 209, 105, '2024-03-09', 7, 29000.30, 'Cancelled', '2024-03-10'),
13 (1020, 210, 105, '2024-03-10', 4, 16000.30, 'Pending', '2024-03-11');
14
15
```

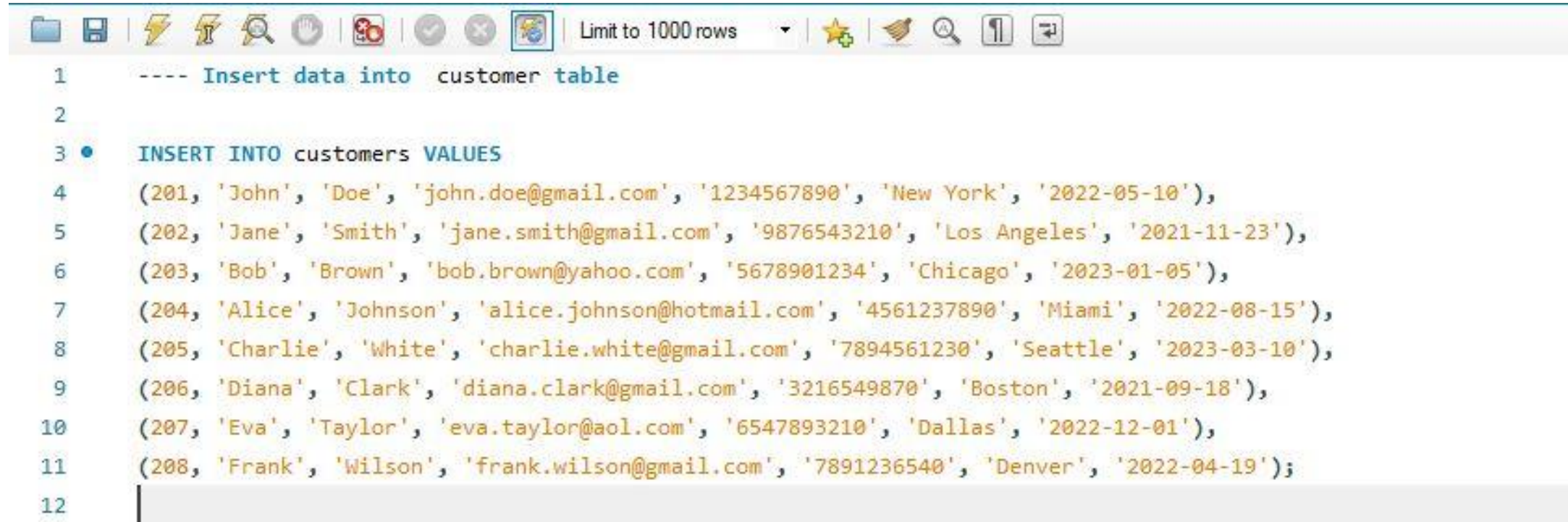
Orders Table View

[illegible]

-- Create customer table

```
1  -- Create customer table
2
3  ● ⊖ CREATE TABLE customers (
4      customer_id INT PRIMARY KEY,
5      first_name VARCHAR(50),
6      last_name VARCHAR(50),
7      email VARCHAR(100),
8      phone VARCHAR(15),
9      city VARCHAR(50),
10     join_date DATE
11 );
12
13 |
```


---- Insert data into customer table



The screenshot shows a SQL IDE window with a toolbar at the top containing icons for file operations, execution, search, and other functions. Below the toolbar, a SQL statement is entered in a text area. The statement is an INSERT INTO query for a table named 'customers'. It contains eight rows of data, each with a unique ID, first name, last name, email, phone number, city, and registration date. The text area has a light blue background and a vertical cursor is visible at the end of the last line. The line numbers 1 through 12 are visible on the left side of the text area.


```
1  ---- Insert data into customer table
2
3  •  INSERT INTO customers VALUES
4      (201, 'John', 'Doe', 'john.doe@gmail.com', '1234567890', 'New York', '2022-05-10'),
5      (202, 'Jane', 'Smith', 'jane.smith@gmail.com', '9876543210', 'Los Angeles', '2021-11-23'),
6      (203, 'Bob', 'Brown', 'bob.brown@yahoo.com', '5678901234', 'Chicago', '2023-01-05'),
7      (204, 'Alice', 'Johnson', 'alice.johnson@hotmail.com', '4561237890', 'Miami', '2022-08-15'),
8      (205, 'Charlie', 'White', 'charlie.white@gmail.com', '7894561230', 'Seattle', '2023-03-10'),
9      (206, 'Diana', 'Clark', 'diana.clark@gmail.com', '3216549870', 'Boston', '2021-09-18'),
10     (207, 'Eva', 'Taylor', 'eva.taylor@aol.com', '6547893210', 'Dallas', '2022-12-01'),
11     (208, 'Frank', 'Wilson', 'frank.wilson@gmail.com', '7891236540', 'Denver', '2022-04-19');
12
```

Customers Table View

[illegible]


-- 1. Find the Total Revenue Generated

```
1  -- 1. Find the Total Revenue Generated
2
3  •  select * from orders;
4  •  select  sum(total_amount) as total_revenue
5
6  from orders
7
8  where order_status = 'delivered' ;
9
```

| | |
|--|---------------|
| < | |
| Result Grid   Filter Rows: <input type="text"/> Export:  Wrap Cell Content:  | |
| | total_revenue |
| ▶ | 133472.03 |

-- 2.) Find the Most Frequently Ordered Product

```
1  -- 2.) Find the Most Frequently Ordered Product
2
3  • select product_id , sum(quantity) as total_quantity
4
5  from orders
6
7  group by product_id
8
9  order by total_quantity desc
10
11 limit 1;
```

| | | |
|--|------------|----------------|
| < | | |
| Result Grid | | |
| Filter Rows: <input type="text"/> | | |
| Export:  | | |
| Wrap Cell Content:  | | |
| Fetch rows:  | | |
| | product_id | total_quantity |
| ▶ | 104 | 25 |

-- 3.) List All Orders That Were Delivered Late

```
1  -- 3.) List All Orders That Were Delivered Late
2
3  • select * from orders;
4
5  • select order_id, shipping_date , order_date
6
7  from orders
8
9  where shipping_date > date_add(order_date, interval 1 day);
```



Result Grid



Filter Rows:

Edit:



Export/Import:



Wrap Cell Content:



| | order_id | shipping_date | order_date |
|---|----------|---------------|------------|
| ▶ | 1015 | 2024-03-06 | 2024-03-04 |
| | 1016 | 2024-03-27 | 2024-03-05 |
| * | NULL | NULL | NULL |

-- 4.) Calculate the Average Order Value for Each Customer

```
1  -- 4.) Calculate the Average Order Value for Each Customer
2
3  • select * from orders;
4  • select * from customers;
5
6  • select customers.first_name, customers.last_name, avg(orders.total_amount) as avg_amount
7
8  from customers
9
10 join orders on customers.customer_id = orders.customer_id
11
12 where orders.order_status= 'delivered'
13
```

Result Grid |   Filter Rows: | Export:  | Wrap Cell Content: 

| first_name | last_name | avg_amount |
|------------|-----------|--------------|
| John | Doe | 25670.230000 |
| Jane | Smith | 19700.500000 |
| Bob | Brown | 20100.700000 |
| Charlie | White | 26000.300000 |
| Diana | Clark | 6000.000000 |
| Frank | Wilson | 36000.300000 |

-- 5.) List Customers Who Have Made More Than 1 Order

```
1  -- 5.) List Customers Who Have Made More Than 1 Order
2
3  • select * from orders;
4  • select * from customers;
5
6  • select customers.first_name, customers.last_name , count(orders.order_id) as total_orders
7
8  from customers
9
10 join orders on orders.customer_id = customers.customer_id
11
12 group by customers.first_name, customers.last_name
13
14 having total_orders >1;
```

<

| | | | | |
|--------------------------|---|---|---|--|
| Result Grid |  |  Filter Rows: <input type="text"/> | Export:  | Wrap Cell Content:  |
| <input type="checkbox"/> | first_name | last_name | total_orders | |

