***Project Idea***

**Electricity Bill Management System Proposal**

**Project Overview:**

The **Electricity Bill Management System** aims to streamline the billing process for electricity consumers. It will allow users (both consumers and utility providers) to manage billing information efficiently. The system will track usage, generate bills, and handle payments.

**Features:**

1. **Customer Registration and Profile Management:**
   * Users can register as customers by providing essential details (name, address, contact information).
   * Customers can view and update their profiles.
2. **Meter Management:**
   * Maintain records of electricity meters (installation date, type, location).
   * Associate meters with specific customers.
3. **Billing Cycle:**
   * Define billing cycles (monthly, bi-monthly, etc.).
   * Generate bills based on meter readings and usage during the billing period.
4. **Bill Generation:**
   * Calculate electricity consumption based on meter readings.
   * Generate bills with due dates and payment instructions.
5. **Payment Tracking:**
   * Record payments made by customers.
   * Send reminders for overdue bills.
6. **Usage History:**
   * Allow customers to view their historical usage data.
   * Display trends and patterns over time.
7. **Admin Dashboard:**
   * Admins can manage customer accounts, meters, and billing cycles.
   * Monitor payment status and resolve disputes.

**Database Schema:**

1. **Customers Table:**
   * Customer ID (Primary Key)
   * Name
   * Address
   * Contact Information
2. **Meters Table:**
   * Meter ID (Primary Key)
   * Installation Date
   * Meter Type (e.g., residential, commercial)
   * Customer ID (Foreign Key)
3. **Bills Table:**
   * Bill ID (Primary Key)
   * Customer ID (Foreign Key)
   * Meter ID (Foreign Key)
   * Billing Period (start date, end date)
   * Usage (in kWh)
   * Amount Due
   * Due Date

**Implementation:**

* Use MySQL as the database management system.
* Develop SQL queries for CRUD operations (Create, Read, Update, Delete).
* Create views for reporting (e.g., overdue bills, payment history).

**Next Steps:**

1. **ERD Design:**
   * Create an Entity-Relationship Diagram (ERD) to visualize the relationships between tables.
2. **Table Creation:**
   * Implement the schema in MySQL Workbench.
   * Define tables and their columns.
3. **Data Population:**
   * Insert sample data for testing.
4. **Queries and Reports:**
   * Write SQL queries to retrieve customer information, bill details, and usage history.

**Conclusion:**

The **Electricity Bill Management System** will enhance efficiency, reduce manual errors, and improve customer satisfaction. It will serve as a valuable tool for both consumers and utility providers.

**Solution:**

CREATE DATABASE ElectricityBillManagementSystem;

USE ElectricityBillManagementSystem;

CREATE TABLE Customer (

Customer\_id INT ,

first\_name VARCHAR(20),

last\_name VARCHAR(20),

phone\_no VARCHAR(20),

email VARCHAR(50),

address VARCHAR(100),

primary key(customer\_id)

);

CREATE TABLE Admin (

admin\_id INT PRIMARY KEY,

name VARCHAR(20),

phone\_no VARCHAR(20),

email VARCHAR(50),

address VARCHAR(100),

Customer\_id INT,

FOREIGN KEY (Customer\_id) REFERENCES Customer(Customer\_id)

);

CREATE TABLE Bill (

bill\_id INT PRIMARY KEY,

start\_date DATE,

end\_date Date,

due\_date DATE,

amount\_due DECIMAL,

Bill\_Status VARCHAR(10),

meter\_reading\_start DECIMAL,

meter\_reading\_end DECIMAL,

usage\_units INT,

Customer\_id INT,

FOREIGN KEY (Customer\_id) REFERENCES Customer(Customer\_id)

);

CREATE TABLE Payment (

payment\_id INT PRIMARY KEY,

payment\_date DATE,

payment\_amount DECIMAL,

Bill\_id int,

Customer\_id INT,

FOREIGN KEY (Customer\_id) REFERENCES Customer(Customer\_id),

FOREIGN KEY (Bill\_id) REFERENCES Bill(Bill\_id)

);

create table Meter(

meter\_id int primary key,

connection\_date Date,

meter\_status varchar(20),

meter\_type varchar(20),

reading\_start int,

reading\_end int,

Customer\_id INT,

FOREIGN KEY (Customer\_id) REFERENCES Customer(Customer\_id)

);

**Which customers have unpaid bills?**

SELECT c.first\_name, c.last\_name

FROM Customer c

JOIN Bill b ON c.Customer\_id = b.Customer\_id

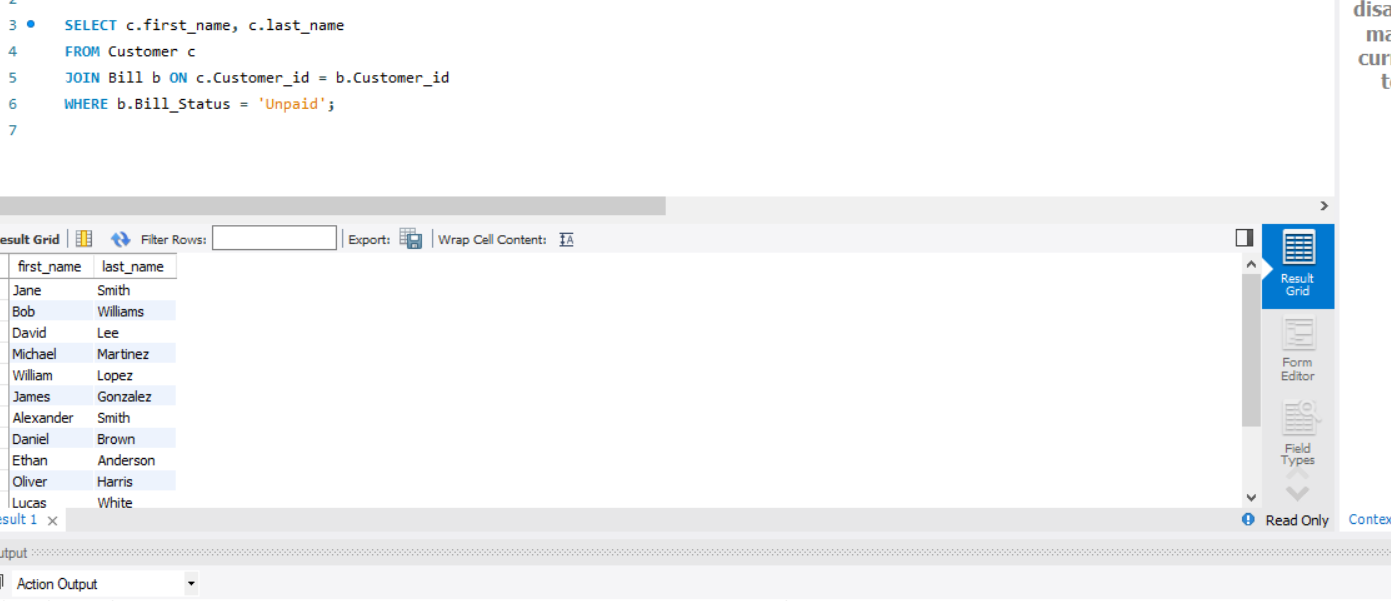
WHERE b.Bill\_Status = 'Unpaid';

SELECT c.first\_name, c.last\_name

FROM Customer c

JOIN Bill b ON c.Customer\_id = b.Customer\_id

WHERE b.Bill\_Status = 'Unpaid';

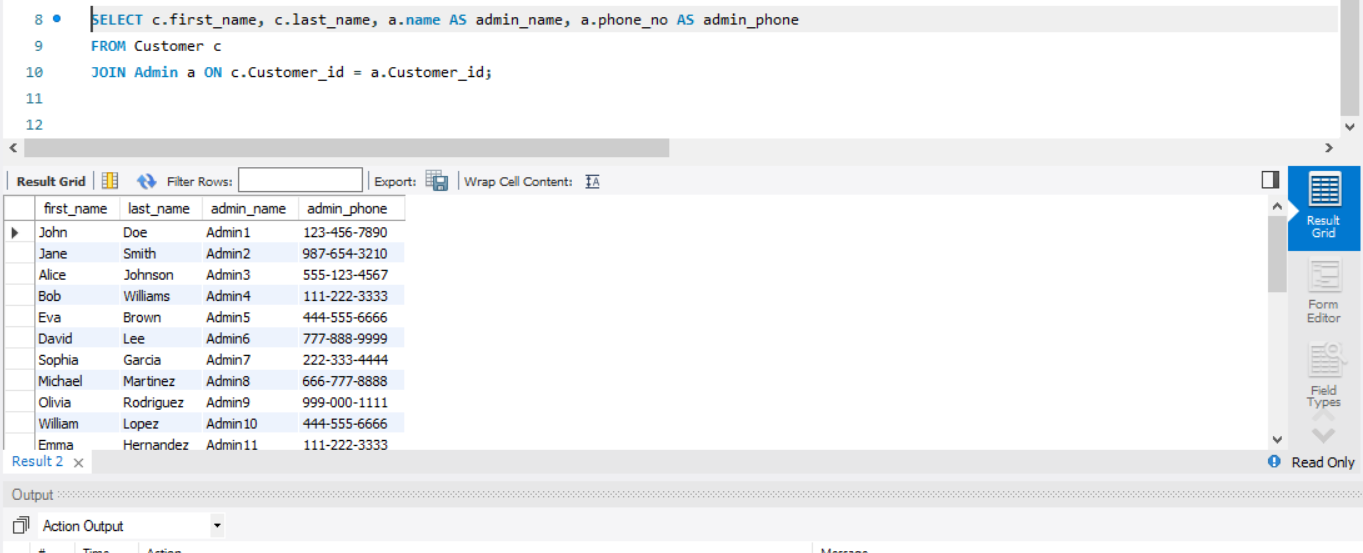


1. **List the names of customers along with their corresponding admin contact details.**

SELECT c.first\_name, c.last\_name, a.name AS admin\_name, a.phone\_no AS admin\_phone

FROM Customer c

JOIN Admin a ON c.Customer\_id = a.Customer\_id;



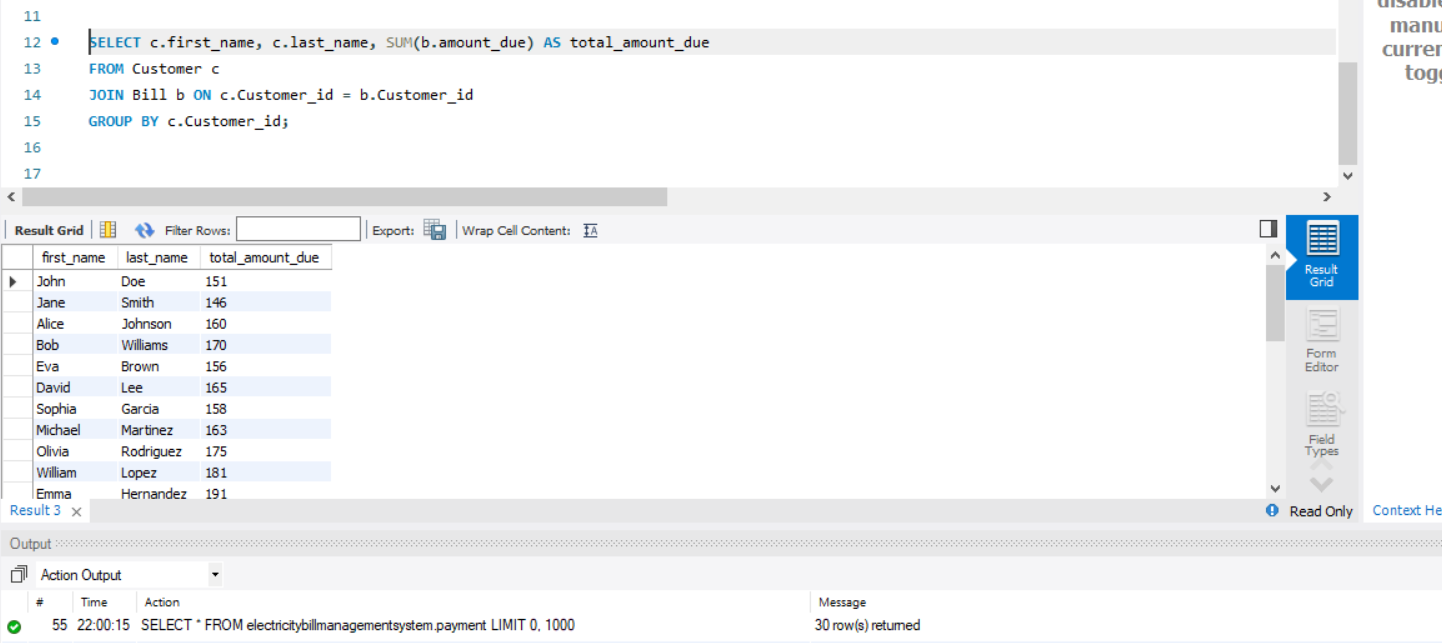
1. **Find the total amount due for each customer.**

SELECT c.first\_name, c.last\_name, SUM(b.amount\_due) AS total\_amount\_due

FROM Customer c

JOIN Bill b ON c.Customer\_id = b.Customer\_id

GROUP BY c.Customer\_id;

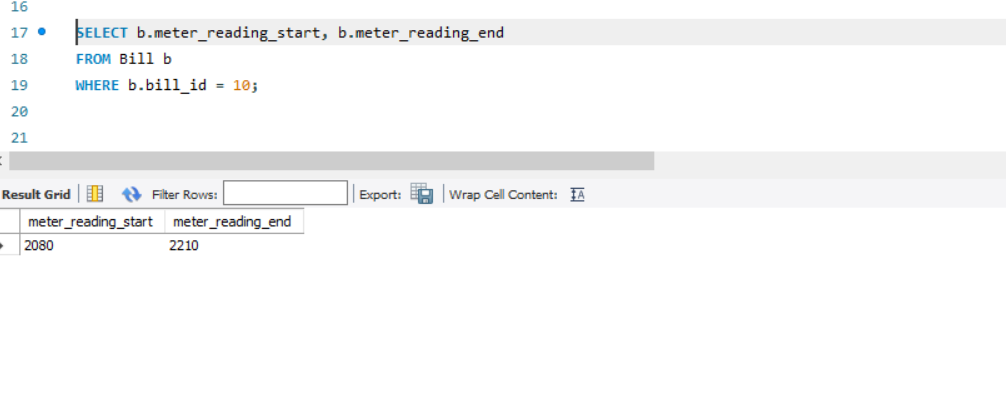


1. **Show the meter readings (start and end) for a specific bill.** Assuming you have a specific bill\_id:

SELECT b.meter\_reading\_start, b.meter\_reading\_end

FROM Bill b

WHERE b.bill\_id = <your\_bill\_id>;



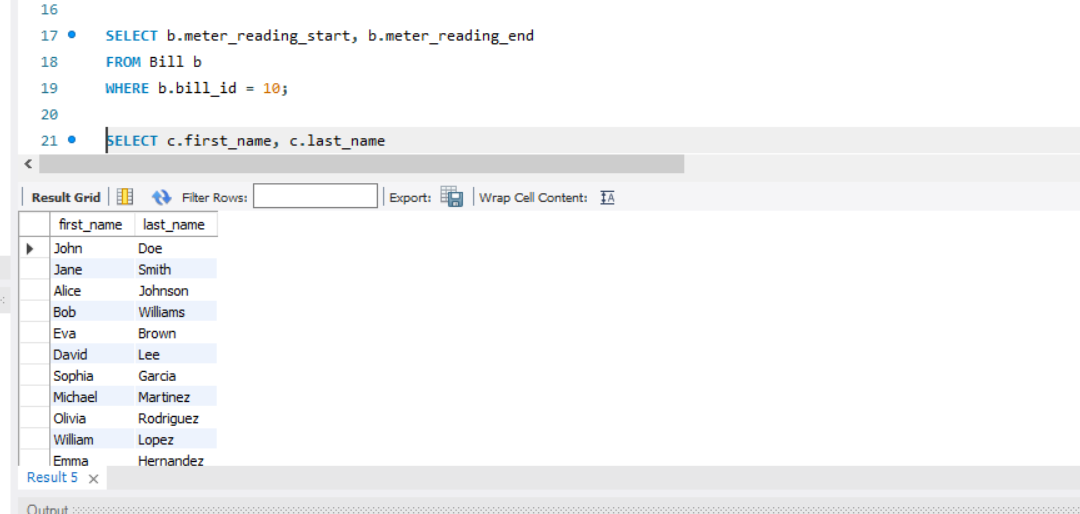
1. **Retrieve the names of customers who have bills with an amount greater than $100.**

SELECT c.first\_name, c.last\_name

FROM Customer c

JOIN Bill b ON c.Customer\_id = b.Customer\_id

WHERE b.amount\_due > 100;



1. **Display the admin details for the customer with the highest total amount due.**

SELECT c.first\_name, c.last\_name, a.name AS admin\_name, a.phone\_no AS admin\_phone

FROM Customer c

JOIN Admin a ON c.Customer\_id = a.Customer\_id

WHERE c.Customer\_id = (

SELECT Customer\_id

FROM (

SELECT Customer\_id, SUM(amount\_due) AS total\_due

FROM Bill

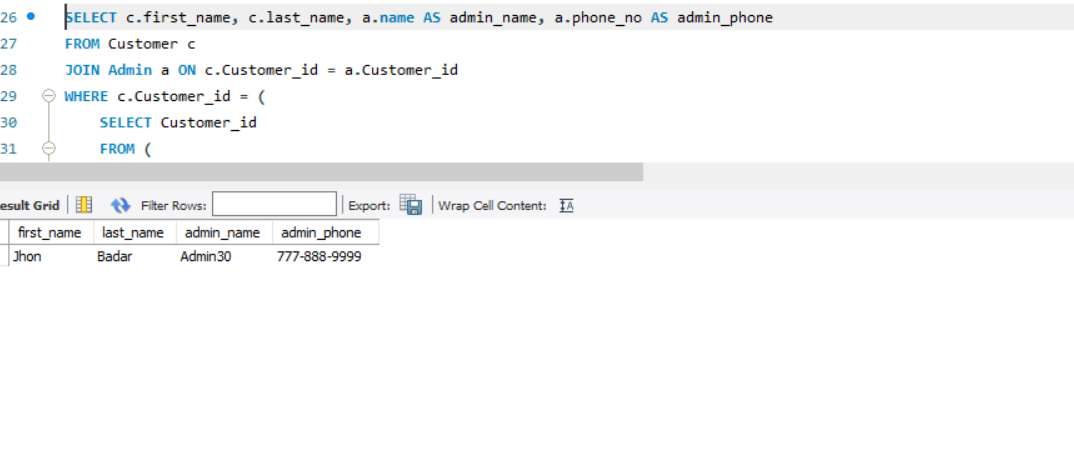
GROUP BY Customer\_id

ORDER BY total\_due DESC

LIMIT 1

) AS subquery

);

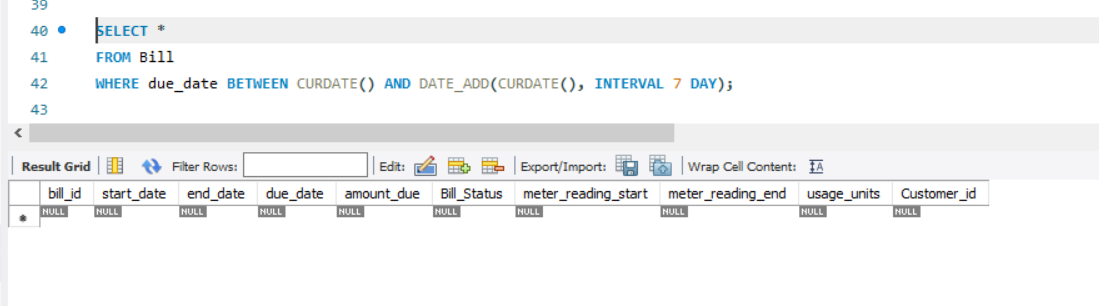


1. **List the bills that have a due date within the next week.**

SELECT \*

FROM Bill

WHERE due\_date BETWEEN CURDATE() AND DATE\_ADD(CURDATE(), INTERVAL 7 DAY);



1. **Show the meter readings for bills with an amount due above the average.**

SELECT b.bill\_id, b.meter\_reading\_start, b.meter\_reading\_end

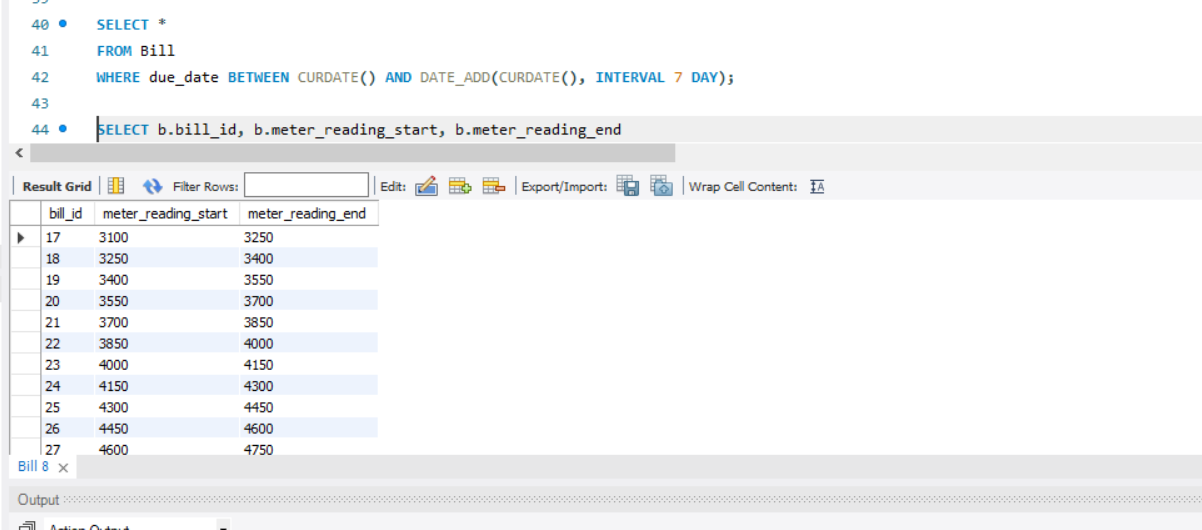
FROM Bill b

WHERE b.amount\_due > (

SELECT AVG(amount\_due)

FROM Bill

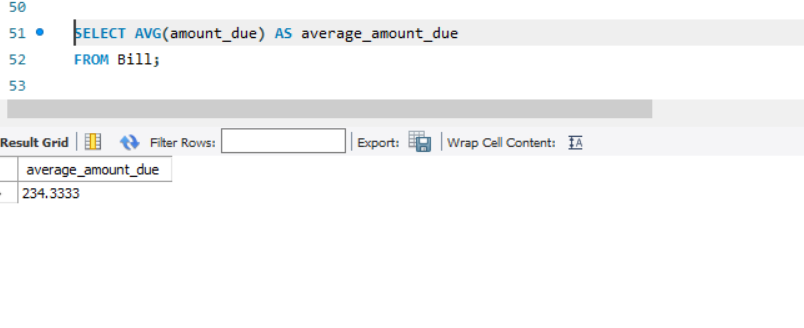
);



1. **Calculate the average amount due for all bills.**

SELECT AVG(amount\_due) AS average\_amount\_due

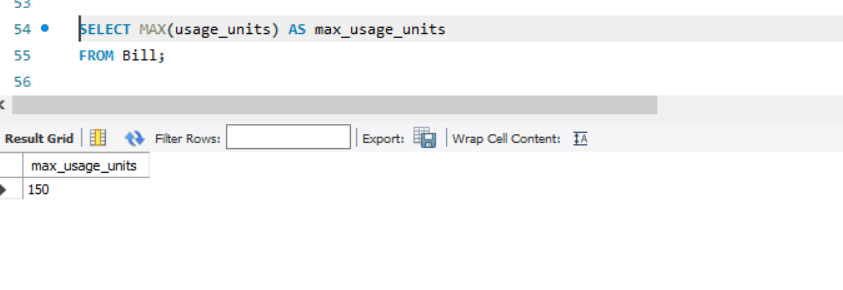
FROM Bill;



1. **Find the maximum usage units among all bills.**

SELECT MAX(usage\_units) AS max\_usage\_units

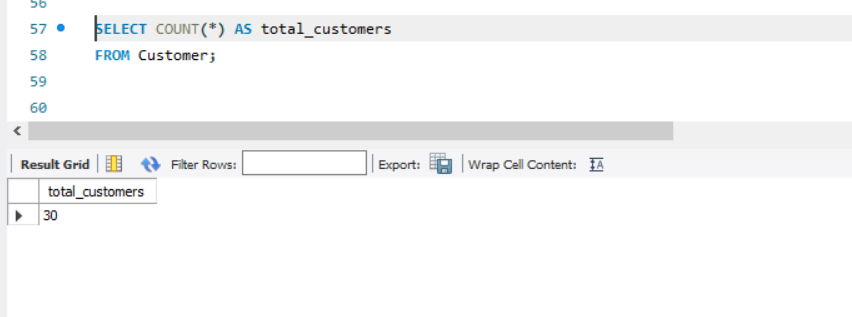
FROM Bill;



1. **Determine the total number of customers in the system.**

SELECT COUNT(\*) AS total\_customers

FROM Customer;



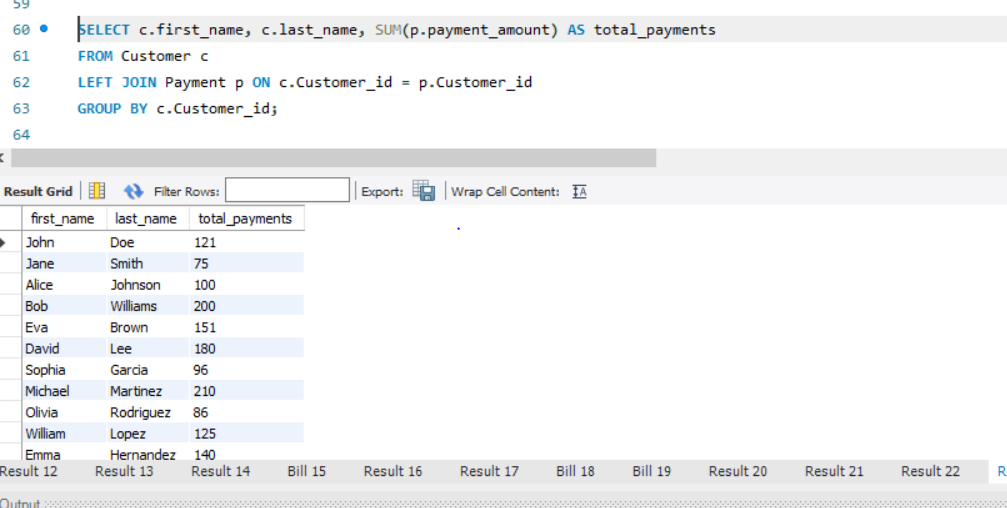
1. **Compute the sum of payments made by each customer.**

SELECT c.first\_name, c.last\_name, SUM(p.payment\_amount) AS total\_payments

FROM Customer c

LEFT JOIN Payment p ON c.Customer\_id = p.Customer\_id

GROUP BY c.Customer\_id;



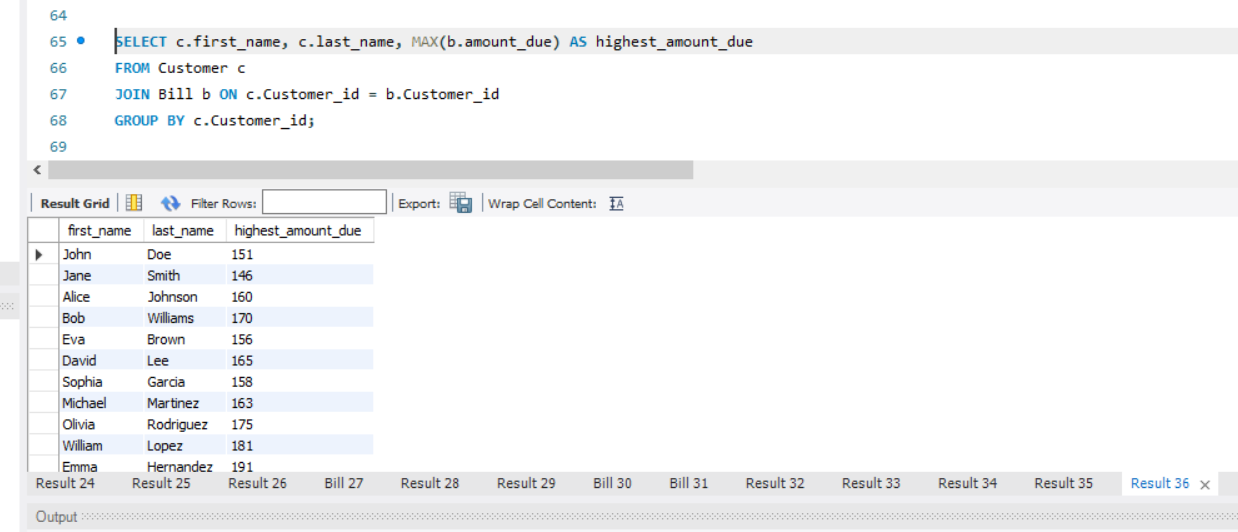
1. **For each customer, find the bill with the highest amount due.**

SELECT c.first\_name, c.last\_name, MAX(b.amount\_due) AS highest\_amount\_due

FROM Customer c

JOIN Bill b ON c.Customer\_id = b.Customer\_id

GROUP BY c.Customer\_id;

1. **Identify customers who have bills with a due date later than their average due date.**

SELECT c.first\_name, c.last\_name

FROM Customer c

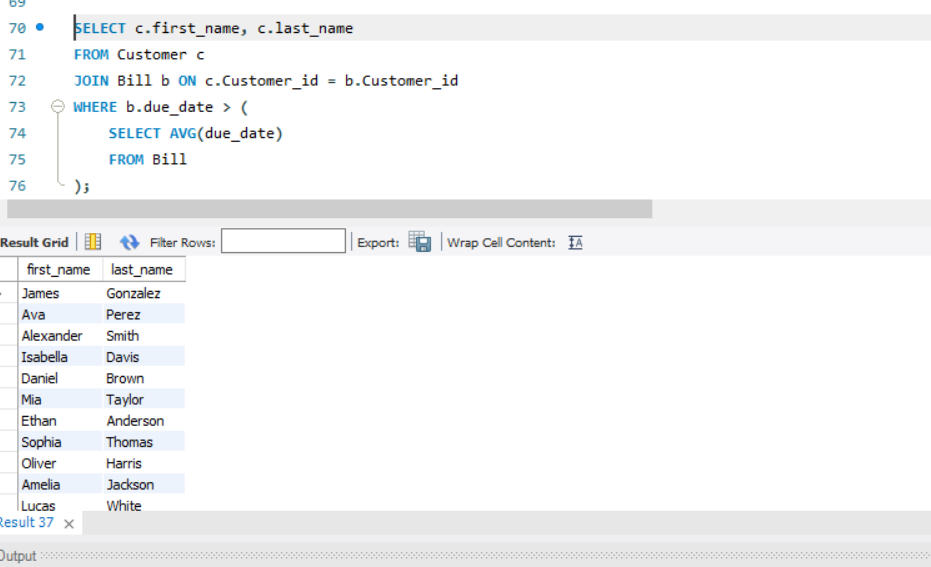
JOIN Bill b ON c.Customer\_id = b.Customer\_id

WHERE b.due\_date > (

SELECT AVG(due\_date)

FROM Bill

);

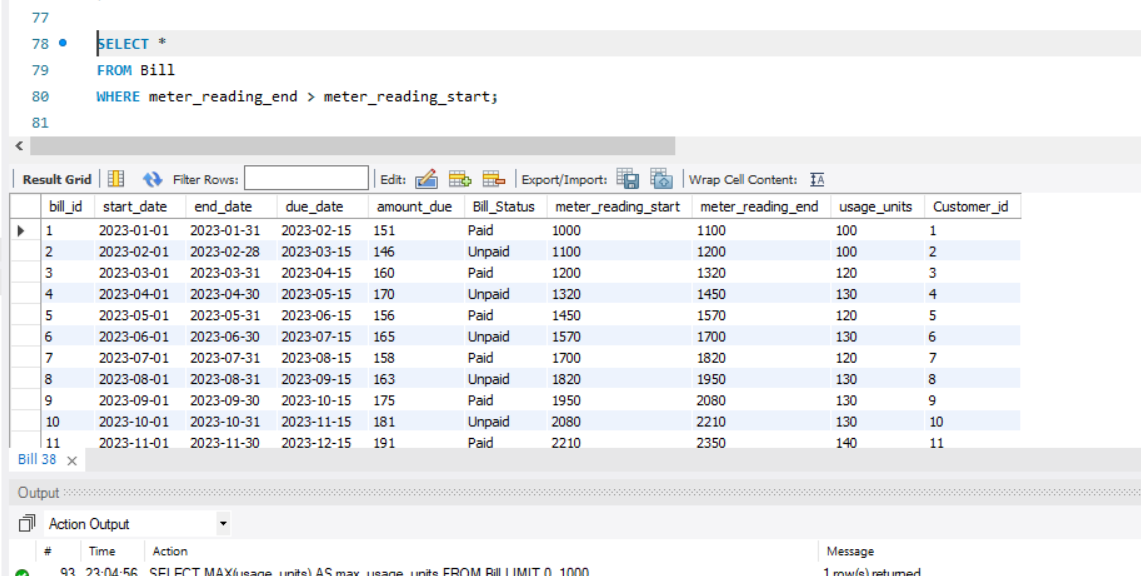


1. **List the bills where the meter reading end value is greater than the start value.**

SELECT \*

FROM Bill

WHERE meter\_reading\_end > meter\_reading\_start;



1. **Show the admin details for the customer with the highest usage units.**

SELECT c.first\_name, c.last\_name, a.name AS admin\_name, a.phone\_no AS admin\_phone

FROM Customer c

JOIN Admin a ON c.Customer\_id = a.Customer\_id

WHERE c.Customer\_id = (

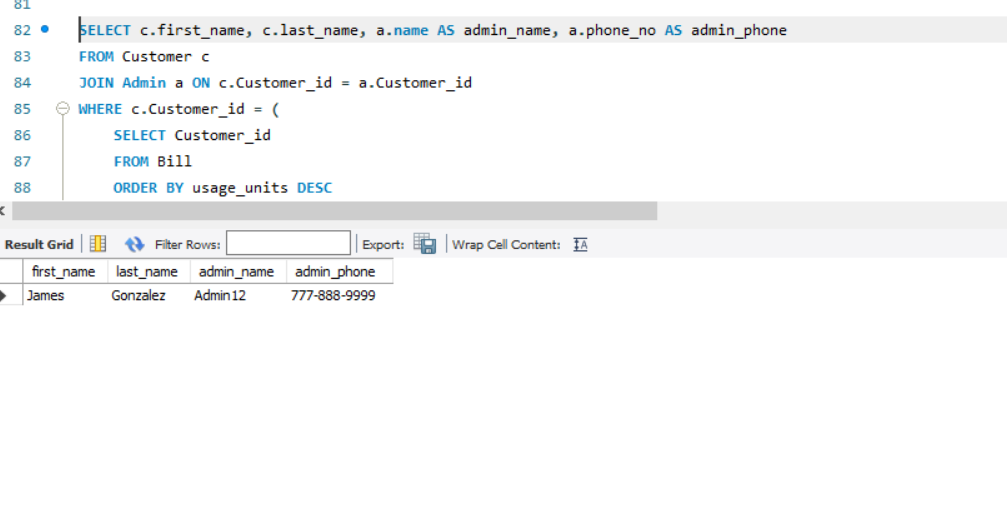
SELECT Customer\_id

FROM Bill

ORDER BY usage\_units DESC

LIMIT 1

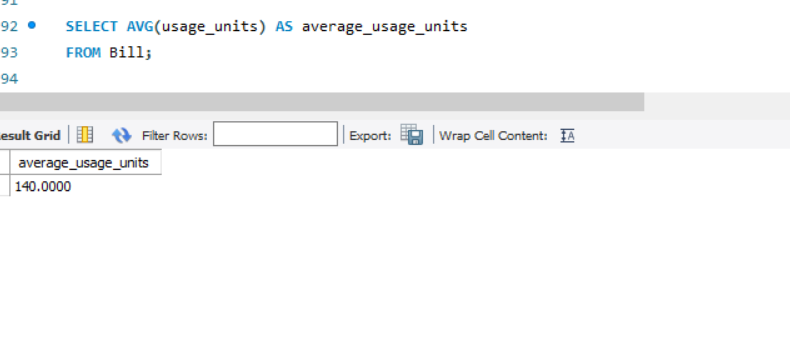
);



1. **Calculate the average usage units for all bills:**

SELECT AVG(usage\_units) AS average\_usage\_units

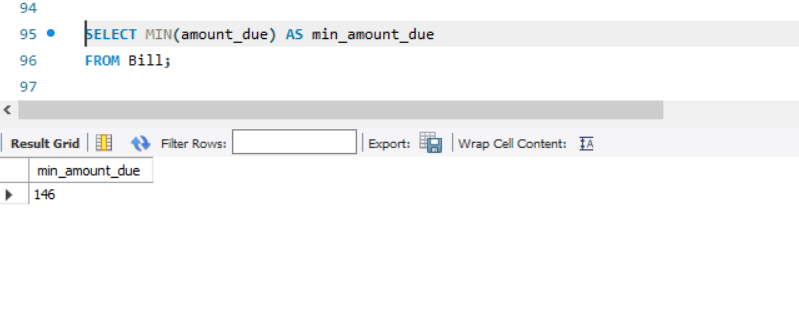
FROM Bill;



1. **Find the minimum amount due among all bills:**

SELECT MIN(amount\_due) AS min\_amount\_due

FROM Bill;



1. **Retrieve the names of customers who have bills with a due date later than their average due date:**

SELECT c.first\_name, c.last\_name

FROM Customer c

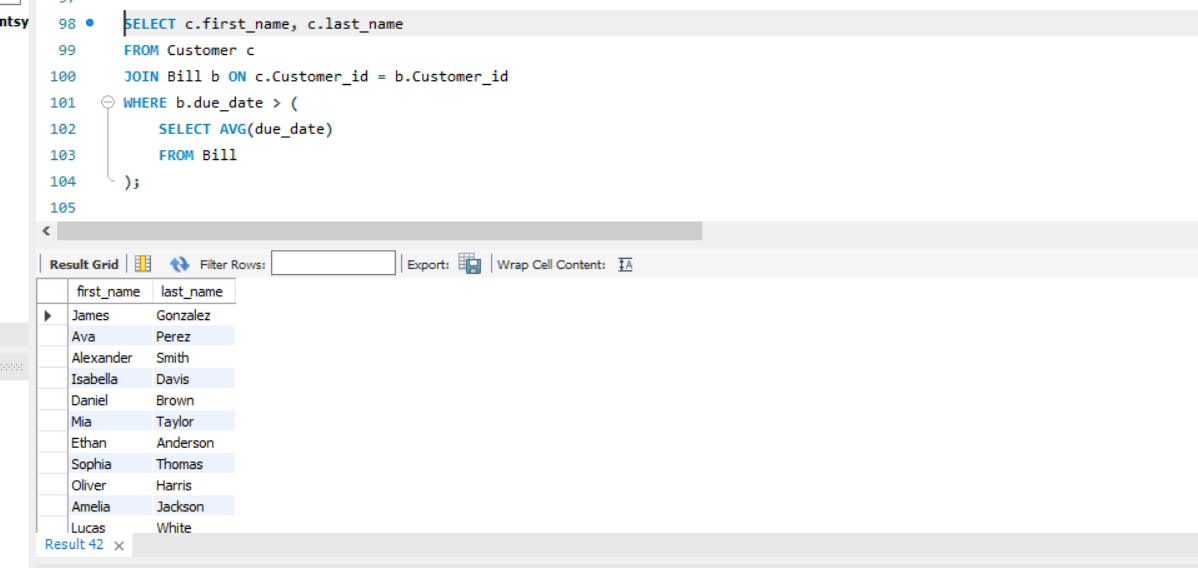
JOIN Bill b ON c.Customer\_id = b.Customer\_id

WHERE b.due\_date > (

SELECT AVG(due\_date)

FROM Bill

);



1. **List the bills where the meter reading end value is greater than the start value:**

SELECT \*

FROM Bill

WHERE meter\_reading\_end > meter\_reading\_start;

