SmallSIEM

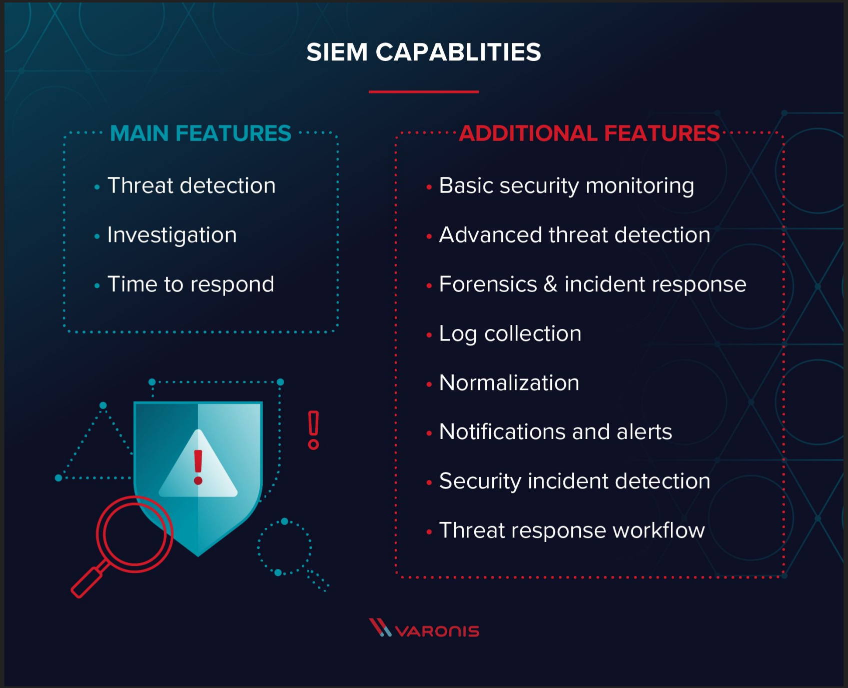
SmallSIEM

By: Thomas Jordan, Thomas Kim, Ewan Shen

Background

Security Information and Event Management

- Field of Cybersecurity
- Software that offers threat detection and incident management
- Uses within an organization



SIEM CAPABLITIES

MAIN FEATURES

- Threat detection
- Investigation
- Time to respond

ADDITIONAL FEATURES

- Basic security monitoring
- Advanced threat detection
- Forensics & incident response
- Log collection
- Normalization
- Notifications and alerts
- Security incident detection
- Threat response workflow

VARONIS

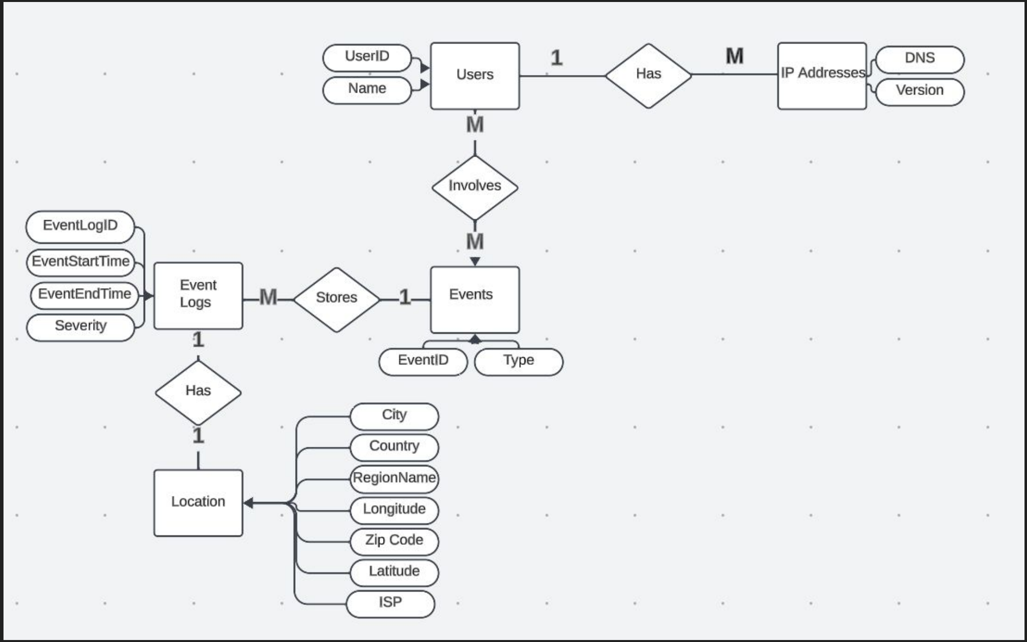
SmallSIEM

SmallSIEM Overview

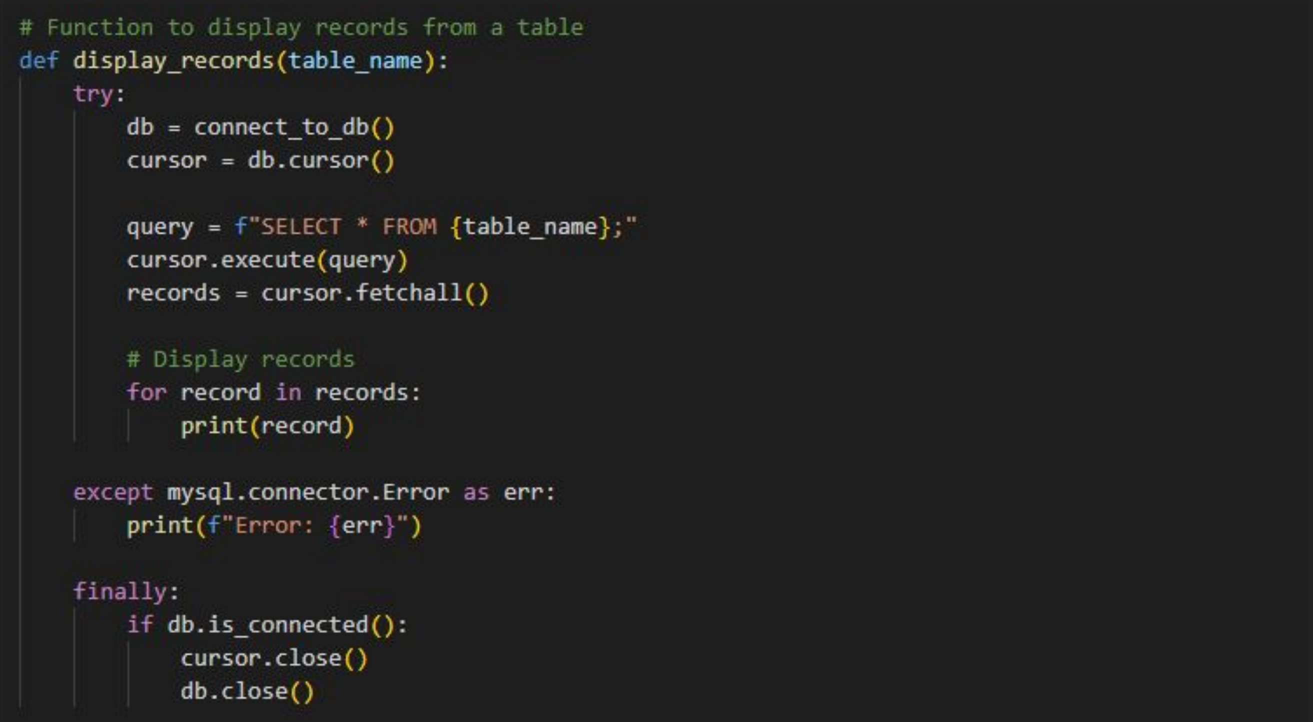
- SmallSIEM supports the storage and interactions between users, IP addresses, locations, and events
- Implements a database
- MySQL, Flask, Jinja2, HTML, Python

Features

- Display records
- Make queries on security events
- Create, delete, and update security events
- Support transactions
- Generate CSV reports of events
- Include views to limit displays of security events to certain users

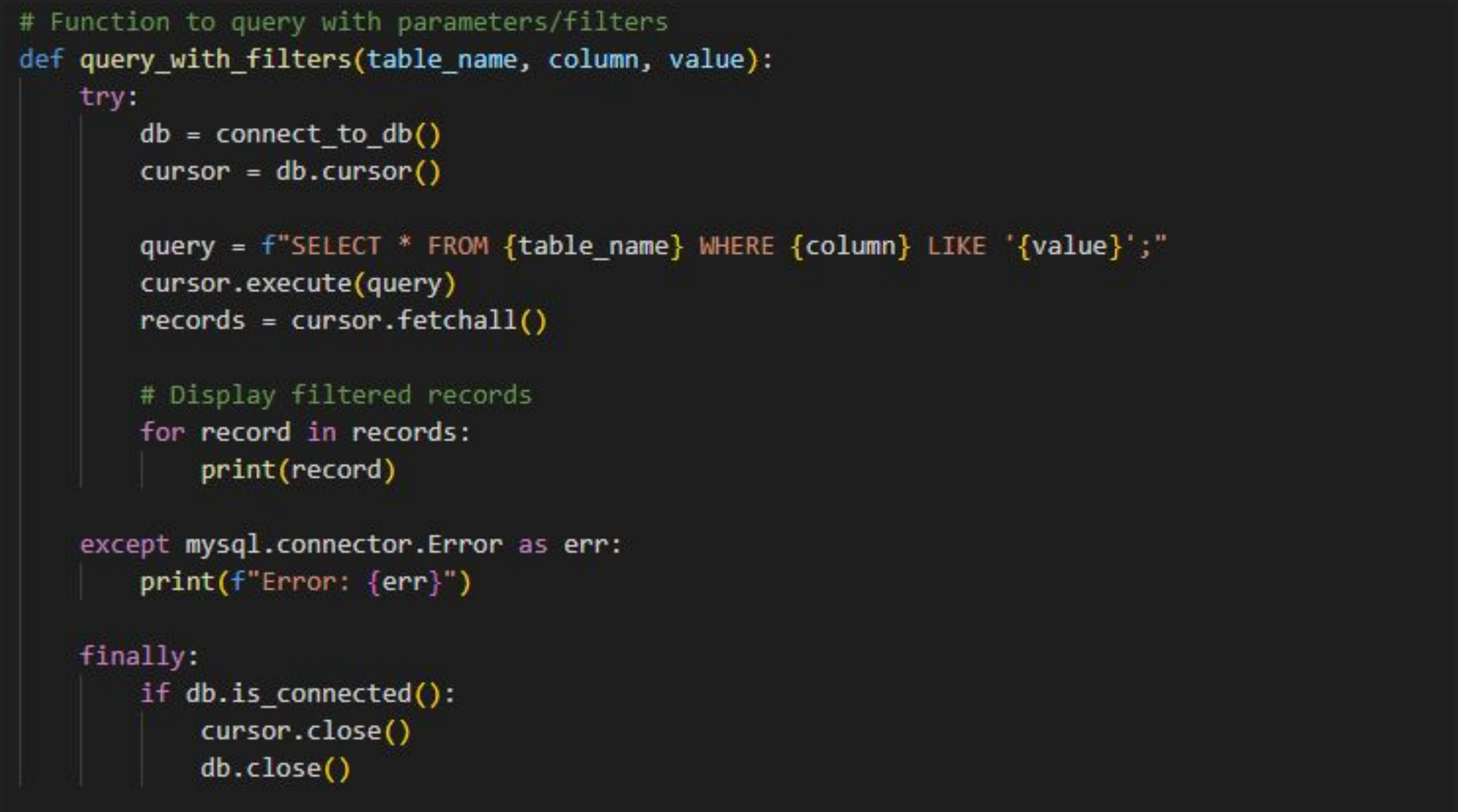


Display Records



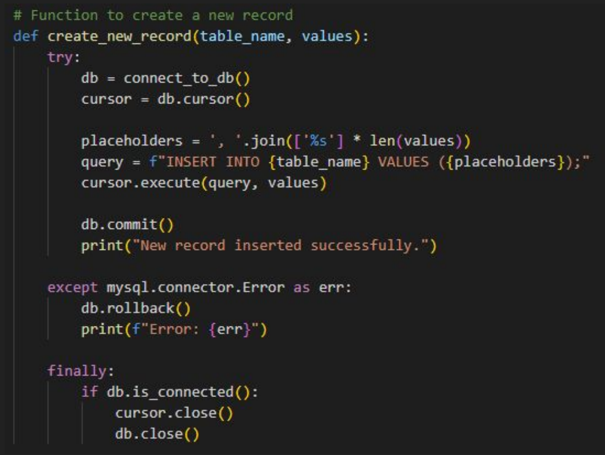
```python
# Function to display records from a table
def display_records(table_name):
    try:
        db = connect_to_db()
        cursor = db.cursor()

        query = f"SELECT * FROM {table_name};"
        cursor.execute(query)
        records = cursor.fetchall()

        # Display records
        for record in records:
            print(record)

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    finally:
        if db.is_connected():
            cursor.close()
            db.close()
```

Make Queries

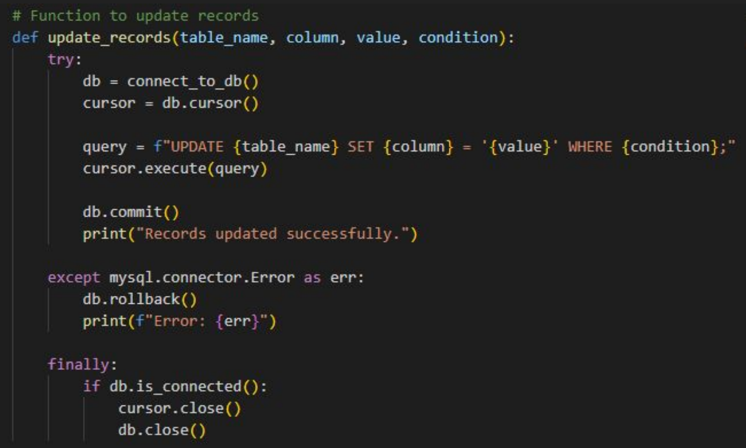


```python
# Function to query with parameters/filters
def query_with_filters(table_name, column, value):
    try:
        db = connect_to_db()
        cursor = db.cursor()

        query = f"SELECT * FROM {table_name} WHERE {column} LIKE '{value}';"
        cursor.execute(query)
        records = cursor.fetchall()

        # Display filtered records
        for record in records:
            print(record)

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    finally:
        if db.is_connected():
            cursor.close()
            db.close()
```

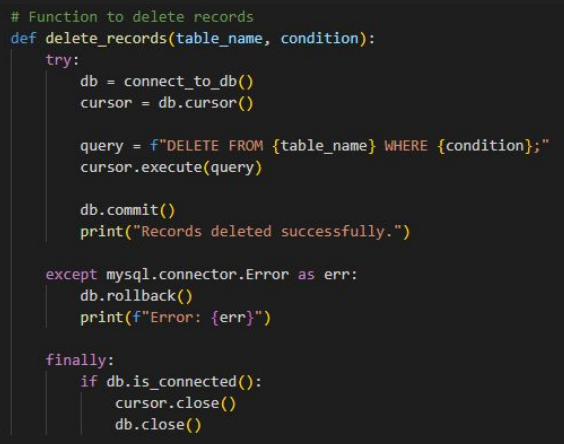
Creating, Updating, Deleting Records



```python
# Function to create a new record
def create_new_record(table_name, values):
    try:
        db = connect_to_db()
        cursor = db.cursor()

        placeholders = ', '.join(['%s'] * len(values))
        query = f"INSERT INTO {table_name} VALUES ({placeholders});"
        cursor.execute(query, values)

        db.commit()
        print("New record inserted successfully.")

    except mysql.connector.Error as err:
        db.rollback()
        print(f"Error: {err}")

    finally:
        if db.is_connected():
            cursor.close()
            db.close()
```

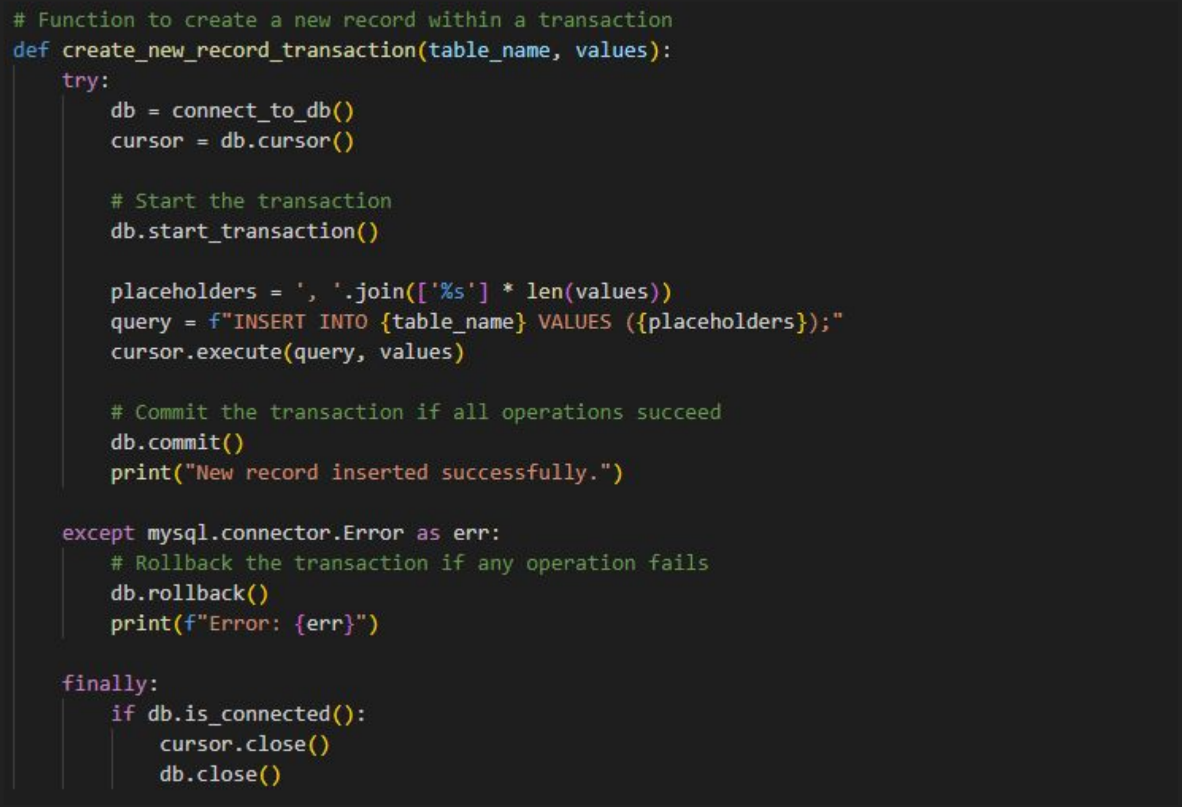


```python
# Function to update records
def update_records(table_name, column, value, condition):
    try:
        db = connect_to_db()
        cursor = db.cursor()

        query = f"UPDATE {table_name} SET {column} = '{value}' WHERE {condition};"
        cursor.execute(query)

        db.commit()
        print("Records updated successfully.")

    except mysql.connector.Error as err:
        db.rollback()
        print(f"Error: {err}")

    finally:
        if db.is_connected():
            cursor.close()
            db.close()
```



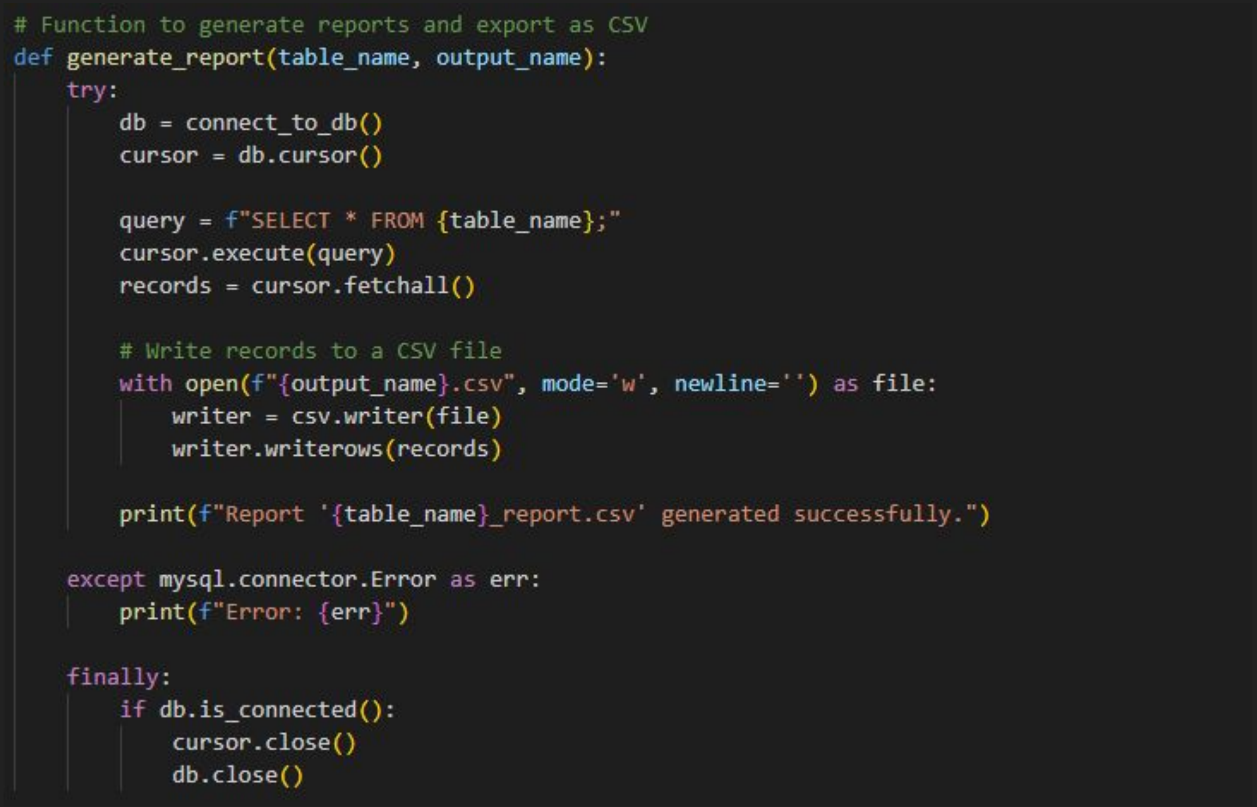
```python
# Function to delete records
def delete_records(table_name, condition):
    try:
        db = connect_to_db()
        cursor = db.cursor()

        query = f"DELETE FROM {table_name} WHERE {condition};"
        cursor.execute(query)

        db.commit()
        print("Records deleted successfully.")

    except mysql.connector.Error as err:
        db.rollback()
        print(f"Error: {err}")

    finally:
        if db.is_connected():
            cursor.close()
            db.close()
```

Transactions



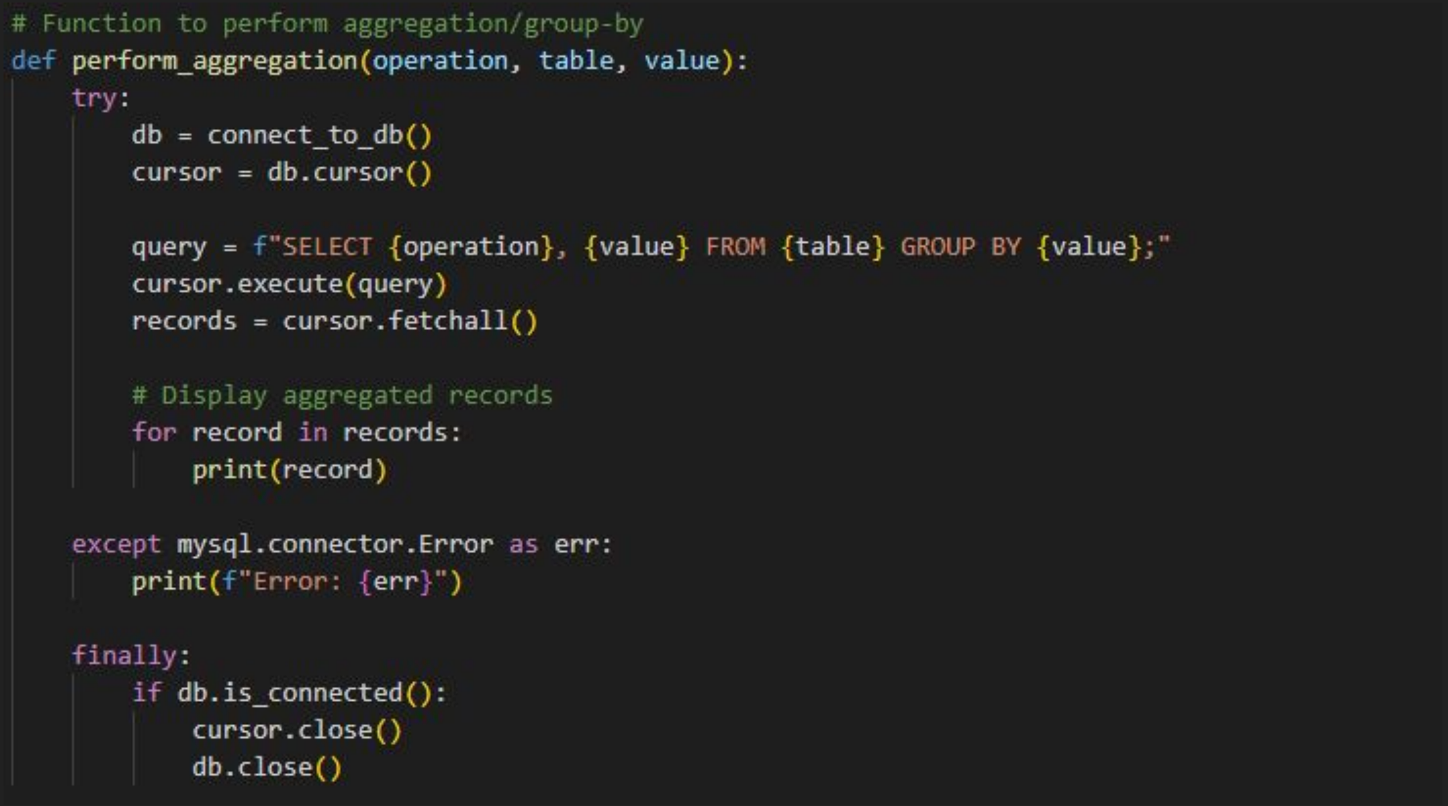
```python
# Function to create a new record within a transaction
def create_new_record_transaction(table_name, values):
    try:
        db = connect_to_db()
        cursor = db.cursor()

        # Start the transaction
        db.start_transaction()

        placeholders = ', '.join(['%s'] * len(values))
        query = f"INSERT INTO {table_name} VALUES ({placeholders});"
        cursor.execute(query, values)

        # Commit the transaction if all operations succeed
        db.commit()
        print("New record inserted successfully.")

    except mysql.connector.Error as err:
        # Rollback the transaction if any operation fails
        db.rollback()
        print(f"Error: {err}")

    finally:
        if db.is_connected():
            cursor.close()
            db.close()
```

Generate CSV Reports



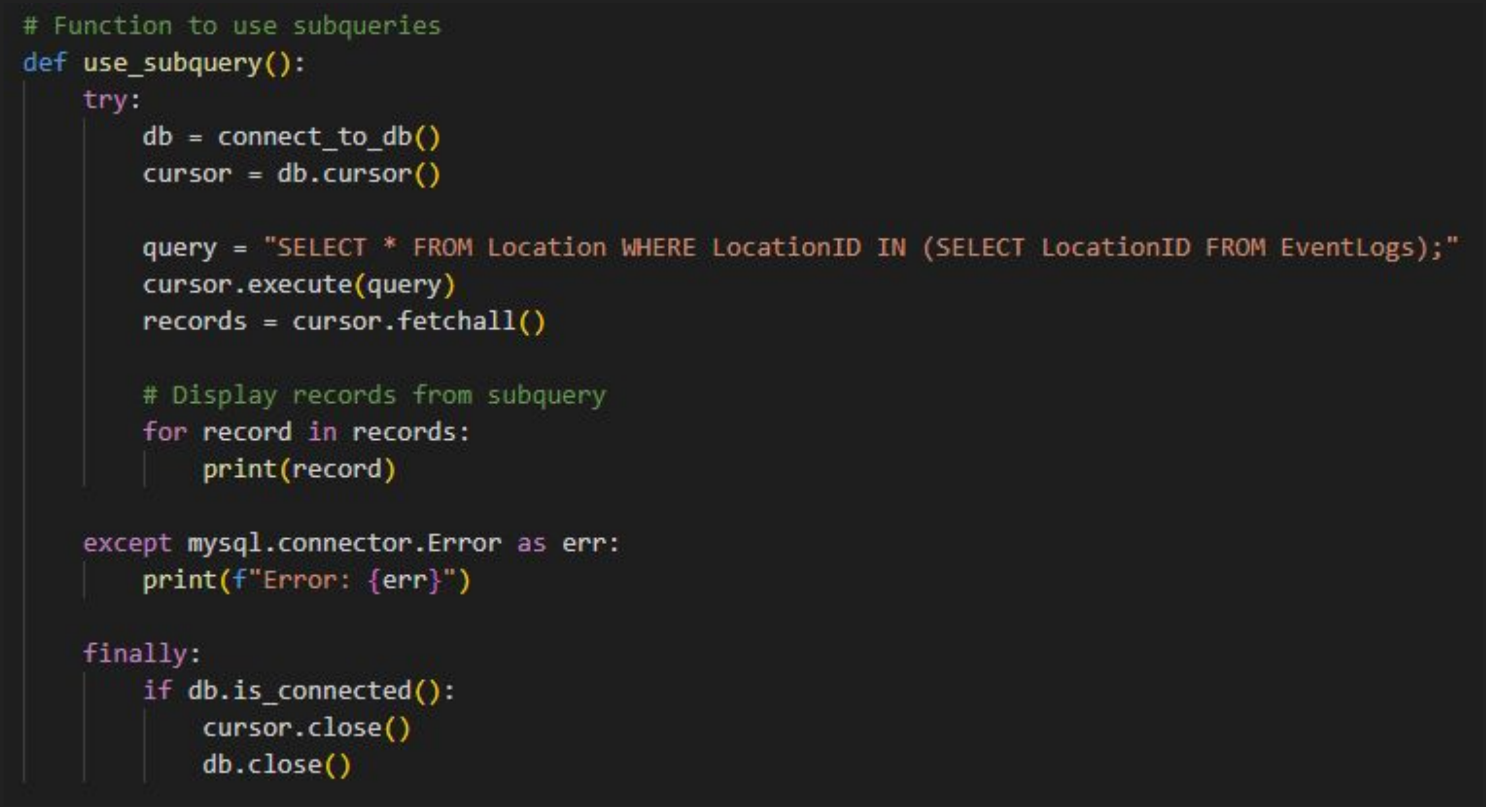
```python
# Function to generate reports and export as CSV
def generate_report(table_name, output_name):
    try:
        db = connect_to_db()
        cursor = db.cursor()

        query = f"SELECT * FROM {table_name};"
        cursor.execute(query)
        records = cursor.fetchall()

        # Write records to a CSV file
        with open(f"{output_name}.csv", mode='w', newline='') as file:
            writer = csv.writer(file)
            writer.writerows(records)

        print(f"Report '{table_name}_report.csv' generated successfully.")

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    finally:
        if db.is_connected():
            cursor.close()
            db.close()
```

Aggregation



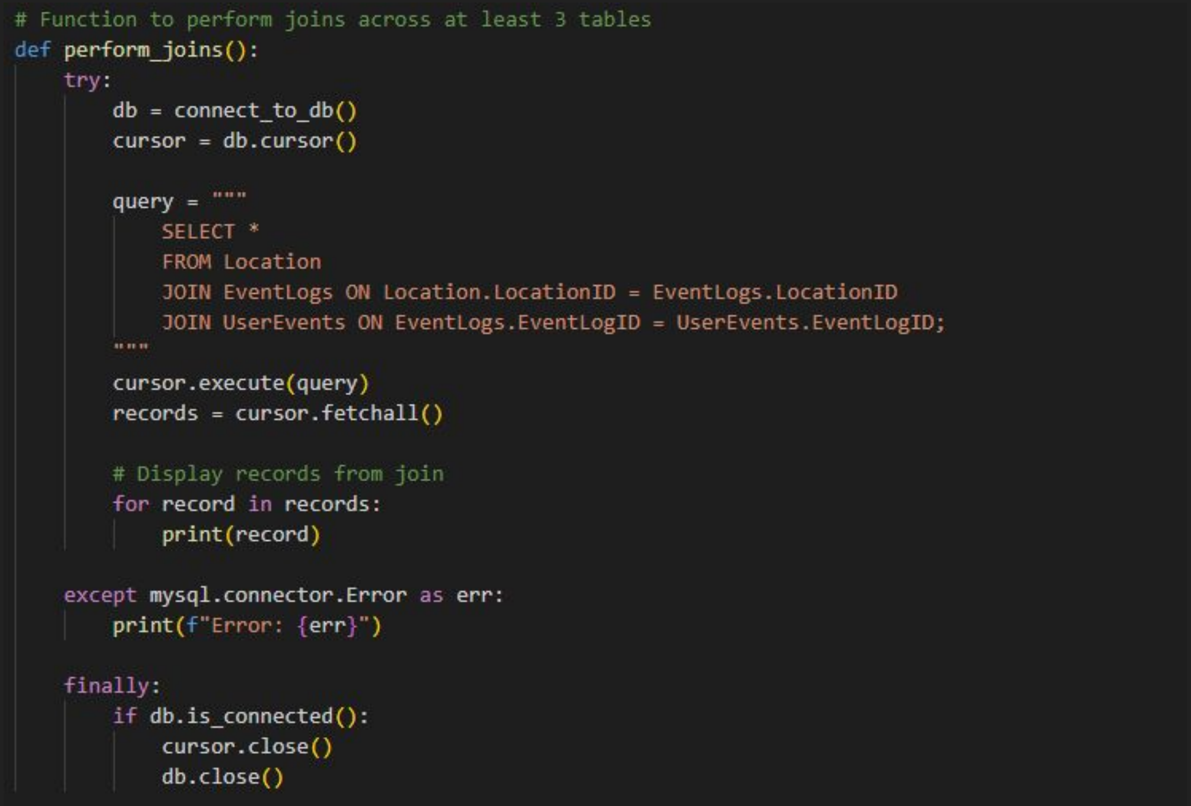
```python
# Function to perform aggregation/group-by
def perform_aggregation(operation, table, value):
    try:
        db = connect_to_db()
        cursor = db.cursor()

        query = f"SELECT {operation}, {value} FROM {table} GROUP BY {value};"
        cursor.execute(query)
        records = cursor.fetchall()

        # Display aggregated records
        for record in records:
            print(record)

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    finally:
        if db.is_connected():
            cursor.close()
            db.close()
```

Subqueries



```python
# Function to use subqueries
def use_subquery():
    try:
        db = connect_to_db()
        cursor = db.cursor()

        query = "SELECT * FROM Location WHERE LocationID IN (SELECT LocationID FROM EventLogs);"
        cursor.execute(query)
        records = cursor.fetchall()

        # Display records from subquery
        for record in records:
            print(record)

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    finally:
        if db.is_connected():
            cursor.close()
            db.close()
```

Joins



```python
# Function to perform joins across at least 3 tables
def perform_joins():
    try:
        db = connect_to_db()
        cursor = db.cursor()

        query = """
            SELECT *
            FROM Location
            JOIN EventLogs ON Location.LocationID = EventLogs.LocationID
            JOIN UserEvents ON EventLogs.EventLogID = UserEvents.EventLogID;
        """
        cursor.execute(query)
        records = cursor.fetchall()

        # Display records from join
        for record in records:
            print(record)

    except mysql.connector.Error as err:
        print(f"Error: {err}")

    finally:
        if db.is_connected():
            cursor.close()
            db.close()
```

DEMO

Remaining Features

- Implementing views
- Reducing the amount of buttons in the frontend as some do not work
- Improving frontend aesthetics
- Majority of remaining features regard the frontend