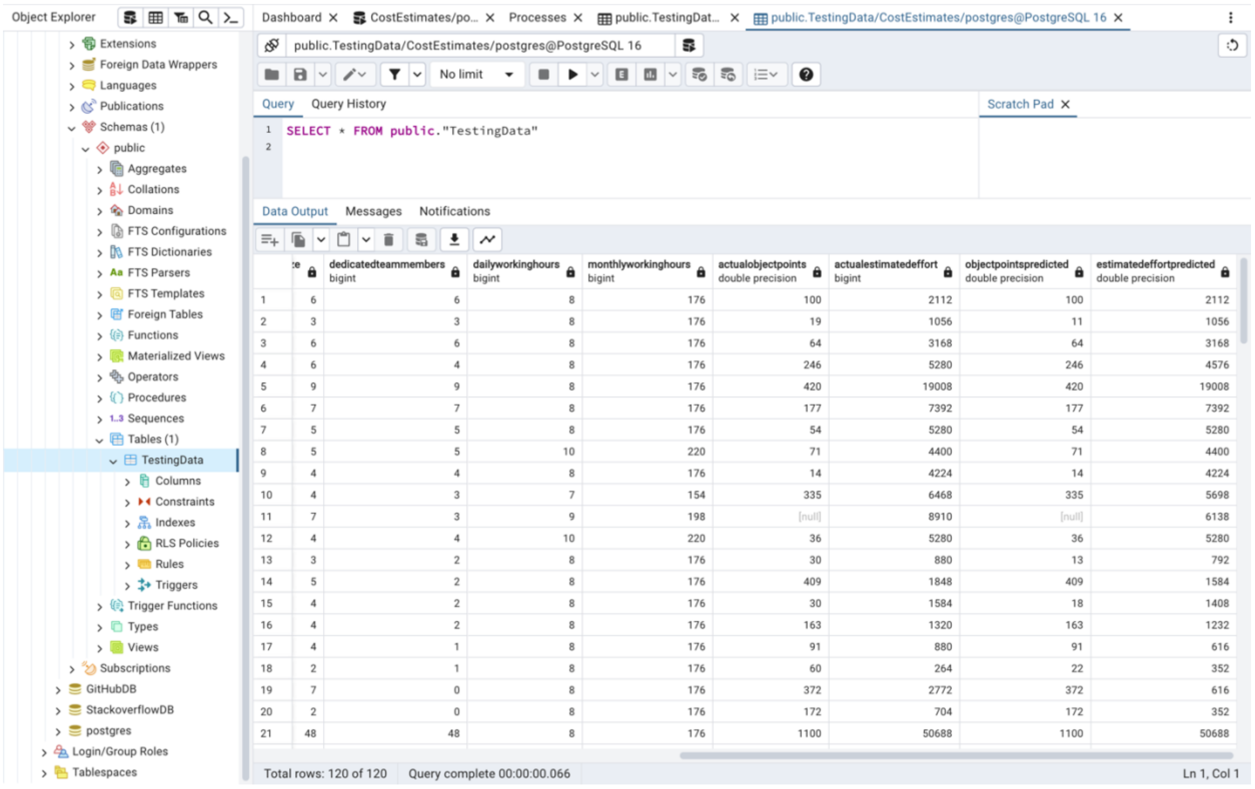
Group 17

The use of generative AI to generate estimates for the project plan, design document, source code, test plan and test cases, and user API documentation.

Kamble Soham skamble2@hawk.iit.edu <A20517098>
Chawla Bhavya bchawla@hawk.iit.edu <A20516957>

Results and Outputs

1. "objectpointspredicted" and "estimatedeffortpredicted" columns are added to the database.

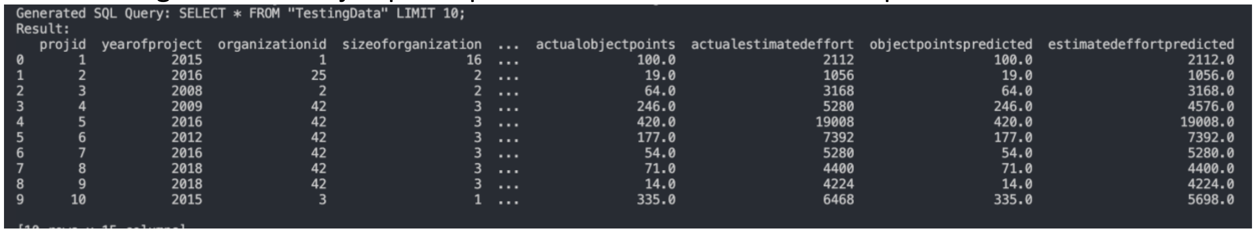


```
SELECT * FROM public."TestingData"
```

	ze	dedicatedteammembers bigint	dailyworkinghours bigint	monthlyworkinghours bigint	actualobjectpoints double precision	actualestimatedeffort bigint	objectpointspredicted double precision	estimatedeffortpredicted double precision
1	6	6	8	176	100	2112	100	2112
2	3	3	8	176	19	1056	11	1056
3	6	6	8	176	64	3168	64	3168
4	6	4	8	176	246	5280	246	4576
5	9	9	8	176	420	19008	420	19008
6	7	7	8	176	177	7392	177	7392
7	5	5	8	176	54	5280	54	5280
8	5	5	10	220	71	4400	71	4400
9	4	4	8	176	14	4224	14	4224
10	4	3	7	154	335	6468	335	5698
11	7	3	9	198	[null]	8910	[null]	6138
12	4	4	10	220	36	5280	36	5280
13	3	2	8	176	30	880	13	792
14	5	2	8	176	409	1848	409	1584
15	4	2	8	176	30	1584	18	1408
16	4	2	8	176	163	1320	163	1232
17	4	1	8	176	91	880	91	616
18	2	1	8	176	60	264	22	352
19	7	0	8	176	372	2772	372	616
20	2	0	8	176	172	704	172	352
21	48	48	8	176	1100	50688	1100	50688

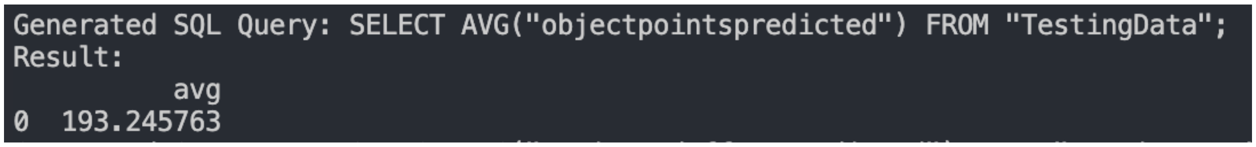
Total rows: 120 of 120 Query complete 00:00:00.066 Ln 1, Col 1

2. Result of the 1st generated SQL query to view the first 10 rows of the TestingData Table along with the "objectpointspredicted" and "estimatedeffortpredicted" columns.



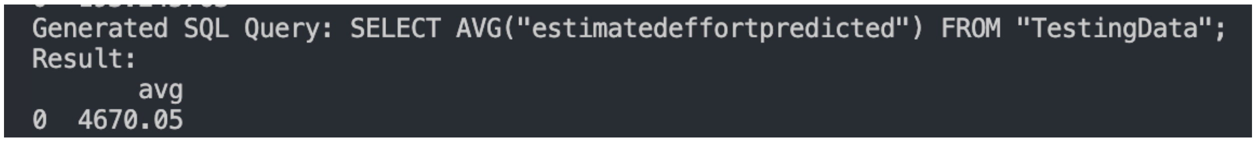
```
Generated SQL Query: SELECT * FROM "TestingData" LIMIT 10;
Result:
   projid  yearofproject  organizationid  sizeoforganization  ...  actualobjectpoints  actualestimatedeffort  objectpointspredicted  estimatedeffortpredicted
0       1           2015               1                  16  ...               100.0                   2112                  100.0                    2112.0
1       2           2016              25                   2  ...                19.0                   1056                   19.0                    1056.0
2       3           2008               2                   2  ...                64.0                   3168                   64.0                    3168.0
3       4           2009              42                   3  ...               246.0                   5280                  246.0                    4576.0
4       5           2016              42                   3  ...               420.0                  19008                  420.0                   19008.0
5       6           2012              42                   3  ...               177.0                   7392                  177.0                    7392.0
6       7           2016              42                   3  ...                54.0                   5280                   54.0                    5280.0
7       8           2018              42                   3  ...                71.0                   4400                   71.0                    4400.0
8       9           2018              42                   3  ...                14.0                   4224                   14.0                    4224.0
9      10           2015               3                   1  ...               335.0                   6468                  335.0                    5698.0
```

3. Result of the 2nd generated SQL query to display the Average of the Predicted Object Points.



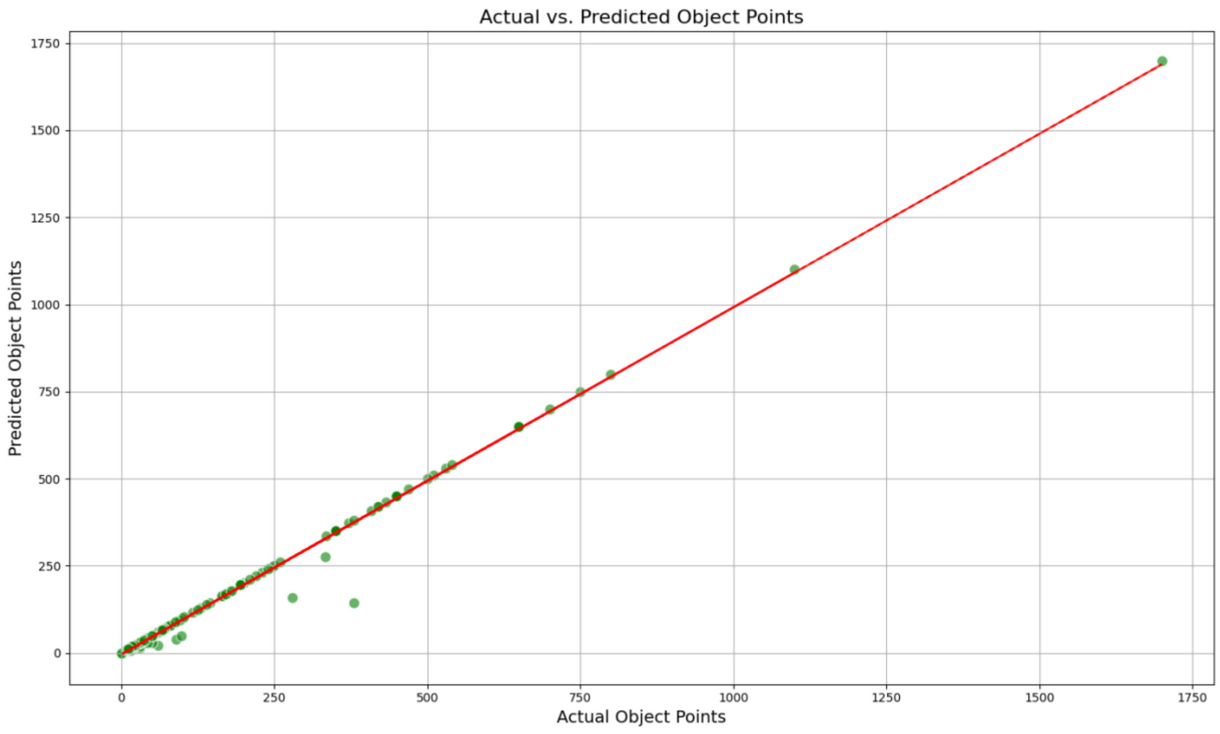
```
Generated SQL Query: SELECT AVG("objectpointspredicted") FROM "TestingData";
Result:
          avg
0  193.245763
```

4. Result of the 3rd generated SQL query to display the Average of the Predicted Estimated Effort.

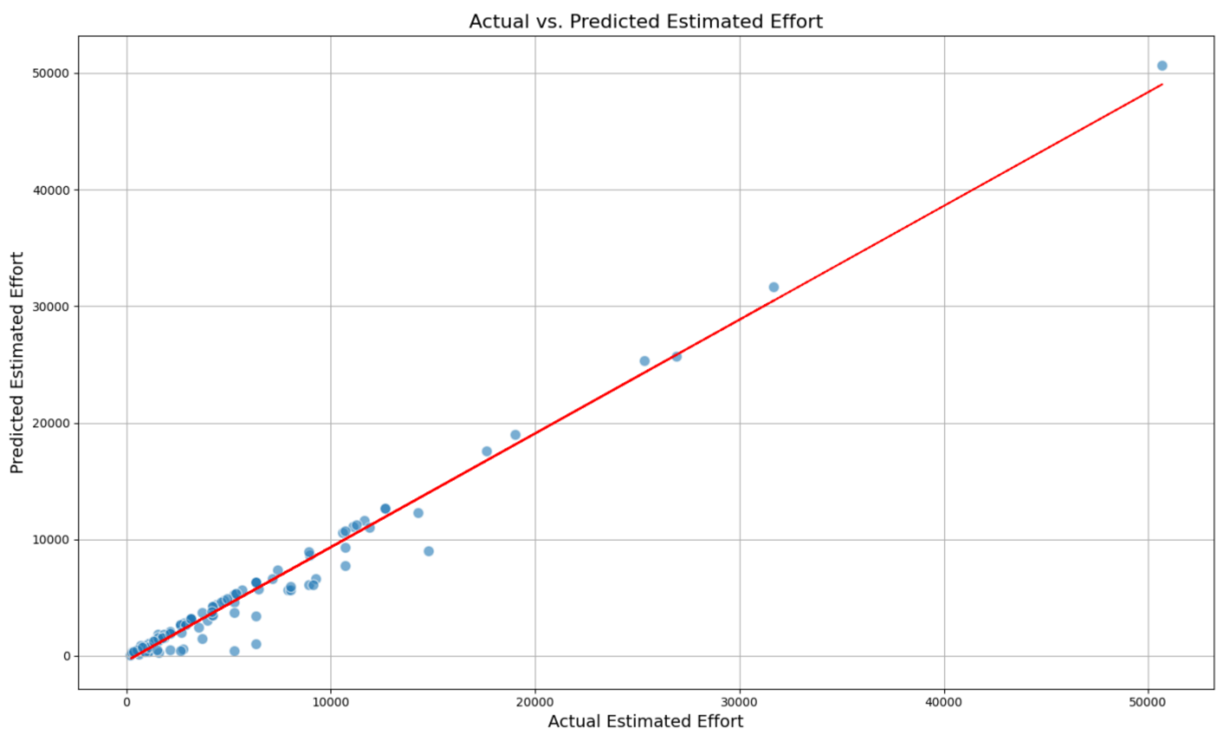


```
Generated SQL Query: SELECT AVG("estimatedeffortpredicted") FROM "TestingData";
Result:
       avg
0  4670.05
```

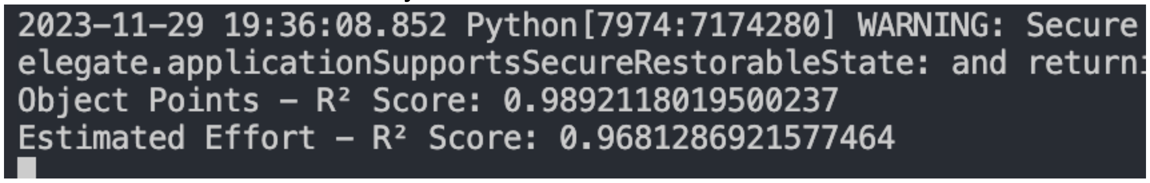
5. Plot of the Actual vs. Predicted Object Points



6. Plot of the Actual vs Predicted Estimated Effort

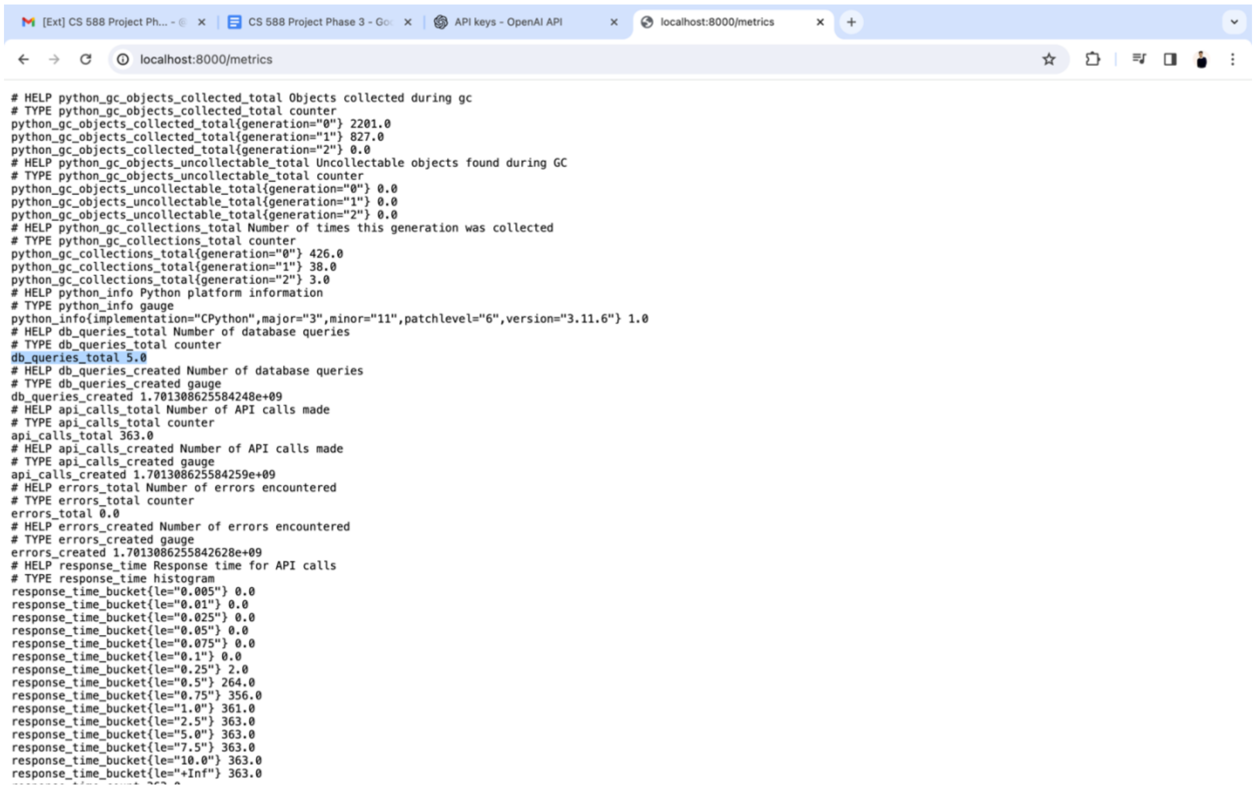


7. The R^2 score of both Object Points and Estimated Effort.



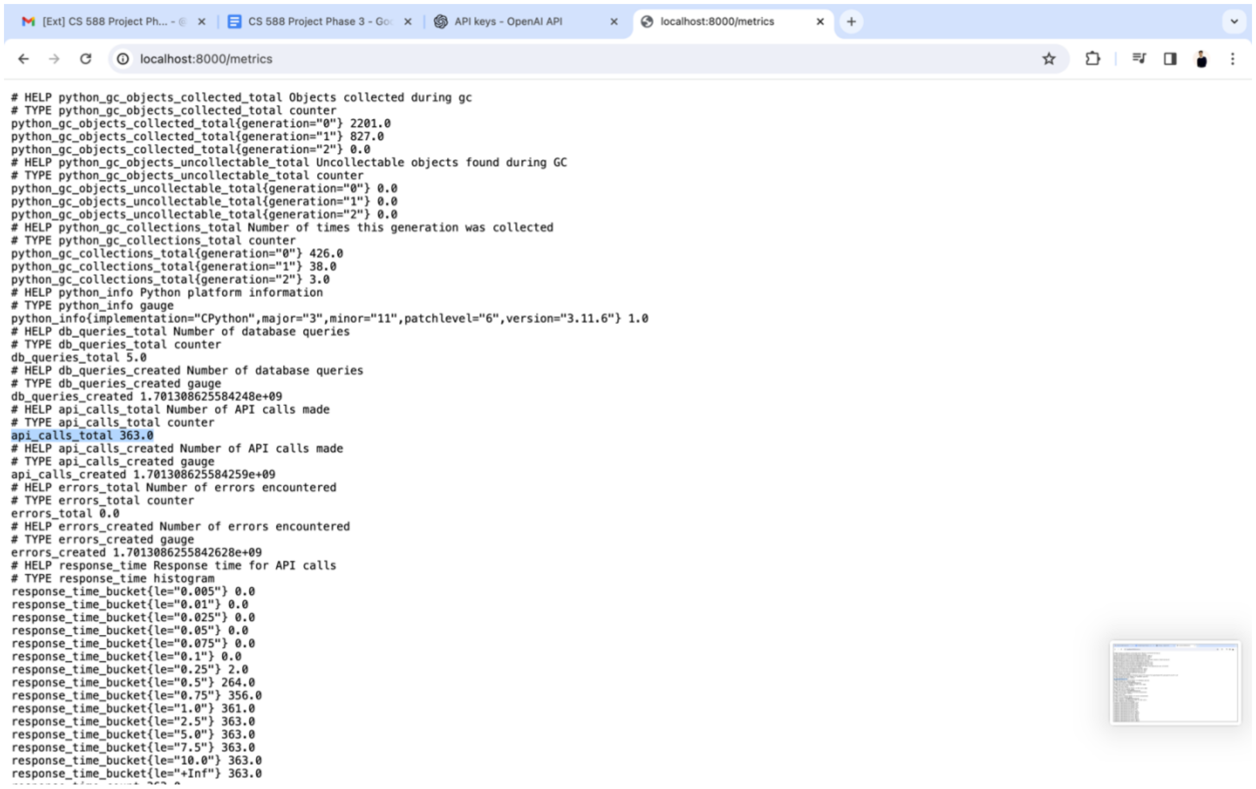
```
2023-11-29 19:36:08.852 Python[7974:7174280] WARNING: Secure
elegate.applicationSupportsSecureRestorableState: and return
Object Points - R² Score: 0.9892118019500237
Estimated Effort - R² Score: 0.9681286921577464
```

8. Total DB queries on Prometheus



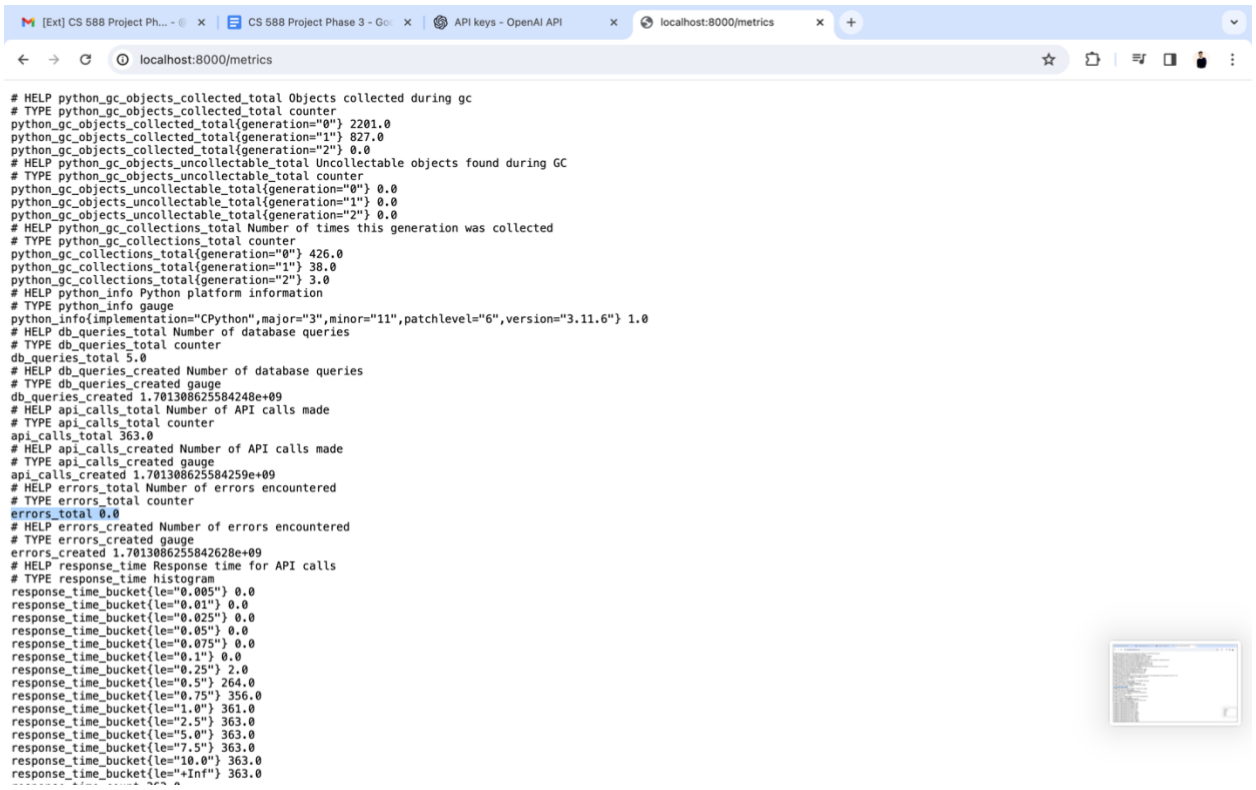
```
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 2201.0
python_gc_objects_collected_total{generation="1"} 827.0
python_gc_objects_collected_total{generation="2"} 0.0
# HELP python_gc_objects_uncollectable_total Uncollectable objects found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 426.0
python_gc_collections_total{generation="1"} 38.0
python_gc_collections_total{generation="2"} 3.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="11",patchlevel="6",version="3.11.6"} 1.0
# HELP db_queries_total Number of database queries
# TYPE db_queries_total counter
db_queries_total 5.0
# HELP db_queries_created Number of database queries
# TYPE db_queries_created gauge
db_queries_created 1.701308625584248e+09
# HELP api_calls_total Number of API calls made
# TYPE api_calls_total counter
api_calls_total 363.0
# HELP api_calls_created Number of API calls made
# TYPE api_calls_created gauge
api_calls_created 1.701308625584259e+09
# HELP errors_total Number of errors encountered
# TYPE errors_total counter
errors_total 0.0
# HELP errors_created Number of errors encountered
# TYPE errors_created gauge
errors_created 1.7013086255842628e+09
# HELP response_time Response time for API calls
# TYPE response_time histogram
response_time_bucket{le="0.005"} 0.0
response_time_bucket{le="0.01"} 0.0
response_time_bucket{le="0.025"} 0.0
response_time_bucket{le="0.05"} 0.0
response_time_bucket{le="0.075"} 0.0
response_time_bucket{le="0.1"} 0.0
response_time_bucket{le="0.25"} 2.0
response_time_bucket{le="0.5"} 264.0
response_time_bucket{le="0.75"} 356.0
response_time_bucket{le="1.0"} 361.0
response_time_bucket{le="2.5"} 363.0
response_time_bucket{le="5.0"} 363.0
response_time_bucket{le="7.5"} 363.0
response_time_bucket{le="10.0"} 363.0
response_time_bucket{le="+Inf"} 363.0
```

9. Total API Calls.



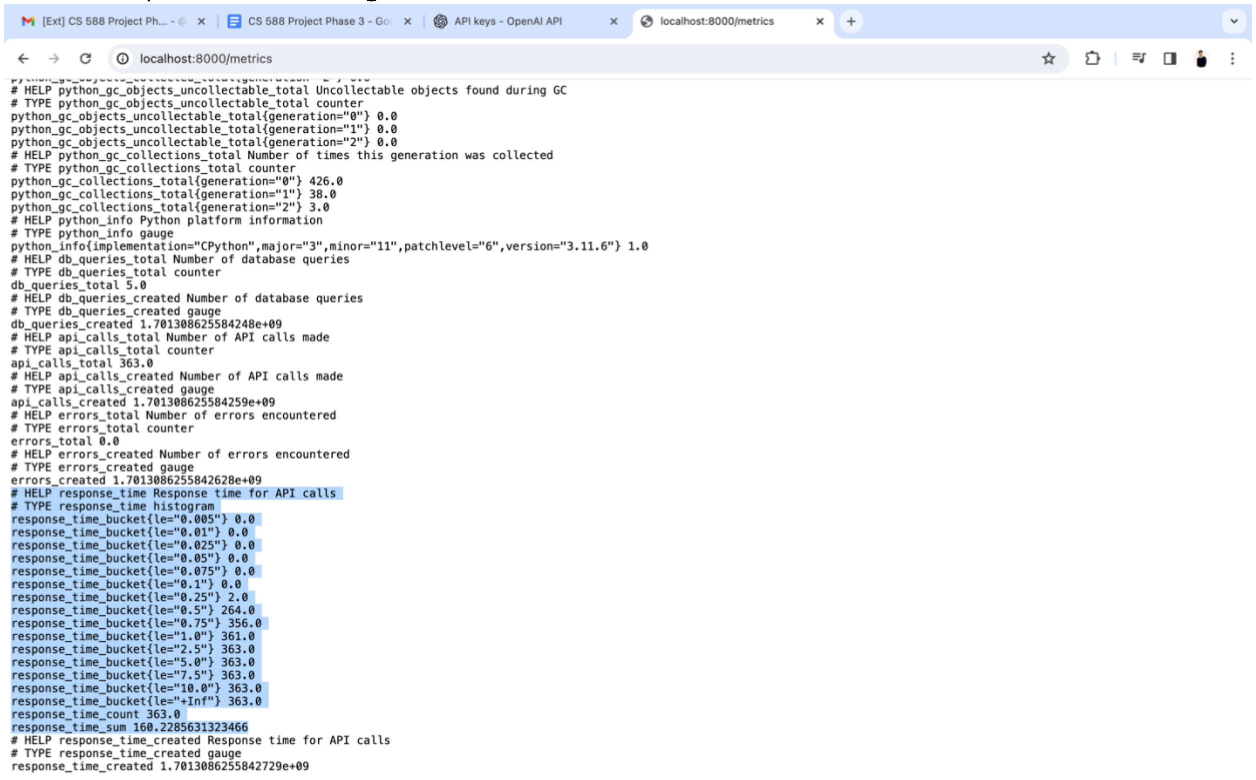
```
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 2201.0
python_gc_objects_collected_total{generation="1"} 827.0
python_gc_objects_collected_total{generation="2"} 0.0
# HELP python_gc_objects_uncollectable_total Uncollectable objects found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 426.0
python_gc_collections_total{generation="1"} 38.0
python_gc_collections_total{generation="2"} 3.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="11",patchlevel="6",version="3.11.6"} 1.0
# HELP db_queries_total Number of database queries
# TYPE db_queries_total counter
db_queries_total 5.0
# HELP db_queries_created Number of database queries
# TYPE db_queries_created gauge
db_queries_created 1.701308625584248e+09
# HELP api_calls_total Number of API calls made
# TYPE api_calls_total counter
api_calls_total 363.0
# HELP api_calls_created Number of API calls made
# TYPE api_calls_created gauge
api_calls_created 1.701308625584259e+09
# HELP errors_total Number of errors encountered
# TYPE errors_total counter
errors_total 0.0
# HELP errors_created Number of errors encountered
# TYPE errors_created gauge
errors_created 1.7013086255842628e+09
# HELP response_time Response time for API calls
# TYPE response_time histogram
response_time_bucket{le="0.005"} 0.0
response_time_bucket{le="0.01"} 0.0
response_time_bucket{le="0.025"} 0.0
response_time_bucket{le="0.05"} 0.0
response_time_bucket{le="0.075"} 0.0
response_time_bucket{le="0.1"} 0.0
response_time_bucket{le="0.25"} 2.0
response_time_bucket{le="0.5"} 264.0
response_time_bucket{le="0.75"} 356.0
response_time_bucket{le="1.0"} 361.0
response_time_bucket{le="2.5"} 363.0
response_time_bucket{le="5.0"} 363.0
response_time_bucket{le="7.5"} 363.0
response_time_bucket{le="10.0"} 363.0
response_time_bucket{le="+Inf"} 363.0
```

10. Total Errors



```
# HELP python_gc_objects_collected_total Objects collected during gc
# TYPE python_gc_objects_collected_total counter
python_gc_objects_collected_total{generation="0"} 2201.0
python_gc_objects_collected_total{generation="1"} 827.0
python_gc_objects_collected_total{generation="2"} 0.0
# HELP python_gc_objects_uncollectable_total Uncollectable objects found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 426.0
python_gc_collections_total{generation="1"} 38.0
python_gc_collections_total{generation="2"} 3.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="11",patchlevel="6",version="3.11.6"} 1.0
# HELP db_queries_total Number of database queries
# TYPE db_queries_total counter
db_queries_total 5.0
# HELP db_queries_created Number of database queries
# TYPE db_queries_created gauge
db_queries_created 1.701308625584248e+09
# HELP api_calls_total Number of API calls made
# TYPE api_calls_total counter
api_calls_total 363.0
# HELP api_calls_created Number of API calls made
# TYPE api_calls_created gauge
api_calls_created 1.701308625584259e+09
# HELP errors_total Number of errors encountered
# TYPE errors_total counter
errors_total 0.0
# HELP errors_created Number of errors encountered
# TYPE errors_created gauge
errors_created 1.7013086255842628e+09
# HELP response_time Response time for API calls
# TYPE response_time histogram
response_time_bucket{le="0.005"} 0.0
response_time_bucket{le="0.01"} 0.0
response_time_bucket{le="0.025"} 0.0
response_time_bucket{le="0.05"} 0.0
response_time_bucket{le="0.075"} 0.0
response_time_bucket{le="0.1"} 0.0
response_time_bucket{le="0.25"} 2.0
response_time_bucket{le="0.5"} 264.0
response_time_bucket{le="0.75"} 356.0
response_time_bucket{le="1.0"} 361.0
response_time_bucket{le="2.5"} 363.0
response_time_bucket{le="5.0"} 363.0
response_time_bucket{le="7.5"} 363.0
response_time_bucket{le="10.0"} 363.0
response_time_bucket{le="+Inf"} 363.0
```

11. Response time Histogram



```
# HELP python_gc_objects_uncollectable_total Uncollectable objects found during GC
# TYPE python_gc_objects_uncollectable_total counter
python_gc_objects_uncollectable_total{generation="0"} 0.0
python_gc_objects_uncollectable_total{generation="1"} 0.0
python_gc_objects_uncollectable_total{generation="2"} 0.0
# HELP python_gc_collections_total Number of times this generation was collected
# TYPE python_gc_collections_total counter
python_gc_collections_total{generation="0"} 426.0
python_gc_collections_total{generation="1"} 38.0
python_gc_collections_total{generation="2"} 3.0
# HELP python_info Python platform information
# TYPE python_info gauge
python_info{implementation="CPython",major="3",minor="11",patchlevel="6",version="3.11.6"} 1.0
# HELP db_queries_total Number of database queries
# TYPE db_queries_total counter
db_queries_total 5.0
# HELP db_queries_created Number of database queries
# TYPE db_queries_created gauge
db_queries_created 1.701308625584248e+09
# HELP api_calls_total Number of API calls made
# TYPE api_calls_total counter
api_calls_total 363.0
# HELP api_calls_created Number of API calls made
# TYPE api_calls_created gauge
api_calls_created 1.701308625584259e+09
# HELP errors_total Number of errors encountered
# TYPE errors_total counter
errors_total 0.0
# HELP errors_created Number of errors encountered
# TYPE errors_created gauge
errors_created 1.7013086255842628e+09
# HELP response_time Response time for API calls
# TYPE response_time histogram
response_time_bucket{le="0.005"} 0.0
response_time_bucket{le="0.01"} 0.0
response_time_bucket{le="0.025"} 0.0
response_time_bucket{le="0.05"} 0.0
response_time_bucket{le="0.075"} 0.0
response_time_bucket{le="0.1"} 0.0
response_time_bucket{le="0.25"} 2.0
response_time_bucket{le="0.5"} 264.0
response_time_bucket{le="0.75"} 356.0
response_time_bucket{le="1.0"} 361.0
response_time_bucket{le="2.5"} 363.0
response_time_bucket{le="5.0"} 363.0
response_time_bucket{le="7.5"} 363.0
response_time_bucket{le="10.0"} 363.0
response_time_bucket{le="+Inf"} 363.0
response_time_count 363.0
response_time_sum 160.2285631323466
# HELP response_time_created Response time for API calls
# TYPE response_time_created gauge
response_time_created 1.7013086255842729e+09
```

Functions and Features.

1. To make connection to the Database

```
connection_string = f"postgresql+psycopg2://{db_user}:{db_password}@{db_host}:{db_port}/{db_name}"


engine = create_engine(connection_string)
```

2. To create counters and histogram variable to store db queries, api calls, errors, and response time in prometheus client and then start the server

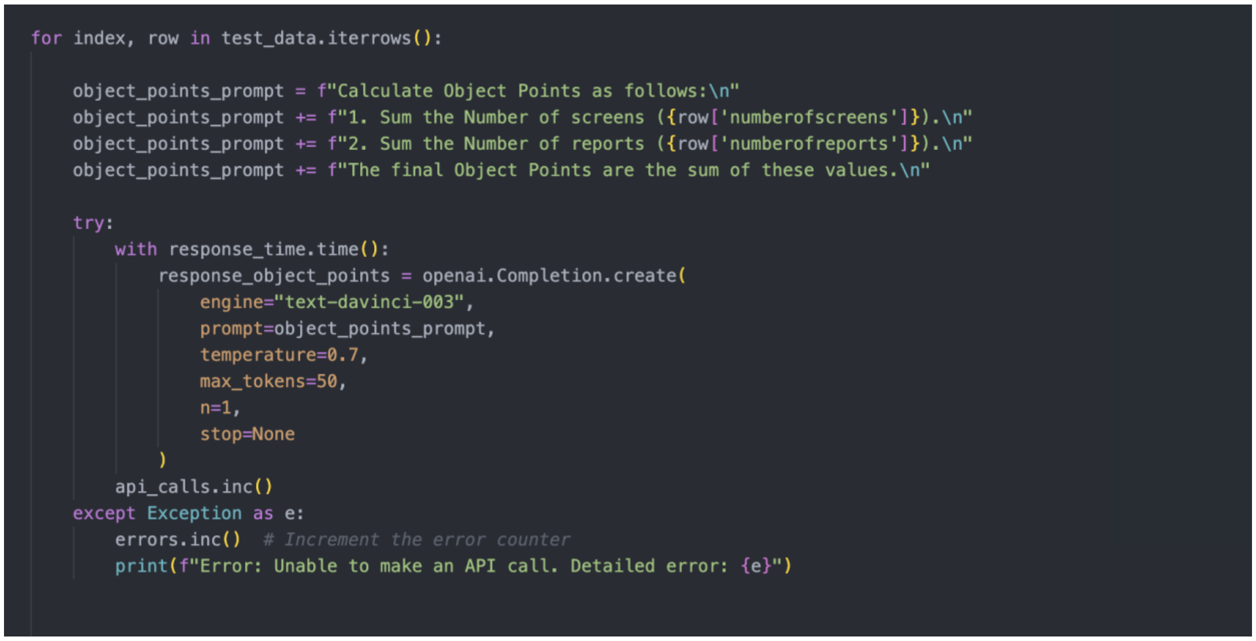
```
db_queries = Counter('db_queries', 'Number of database queries')
api_calls = Counter('api_calls', 'Number of API calls made')
errors = Counter('errors', 'Number of errors encountered')
response_time = Histogram('response_time', 'Response time for API calls')


start_http_server(8000)
```

3. To store the data from the database as a dataframe and to increment the db_queries counter along with exception handling.

```
try:
   test_data = pd.read_sql('SELECT * FROM "TestingData"', engine)
   db_queries.inc()
   print(test_data.columns)
except Exception as e:
   errors.inc()
   print(f"Error: Unable to fetch data from the database. Detailed error: {e}")
```

4. To send a prompt to OpenAI API to calculate the Object Points for each record in the table. Also to increment the api_calls counter along with exception handling.



```
for index, row in test_data.iterrows():

    object_points_prompt = f"Calculate Object Points as follows:\n"
    object_points_prompt += f"1. Sum the Number of screens ({row['numberofscreens']}).\n"
    object_points_prompt += f"2. Sum the Number of reports ({row['numberofreports']}).\n"
    object_points_prompt += f"The final Object Points are the sum of these values.\n"

    try:
        with response_time.time():
            response_object_points = openai.Completion.create(
                engine="text-davinci-003",
                prompt=object_points_prompt,
                temperature=0.7,
                max_tokens=50,
                n=1,
                stop=None
            )
        api_calls.inc()
    except Exception as e:
        errors.inc()  # Increment the error counter
        print(f"Error: Unable to make an API call. Detailed error: {e}")
```

5. To calculate the estimated effort.

```
# Prepare the prompt for the first part of Estimated Effort (Duration * Team members)
  effort_part1_prompt = f"Calculate the first part of Estimated Effort using the formula: Estimated duration ({row['estimatedduration']} days) * Dedicated Team members ({row['dedicatedteammembers']})."

  # Make an API call for the first part of Estimated Effort
  try:
    with response_time.time():
      response_part1 = openai.Completion.create(
        engine="text-davinci-003",
        prompt=effort_part1_prompt,
        temperature=0.7,
        max_tokens=50,
        n=1,
        stop=None
      )
    api_calls.inc()  # Increment the API call counter
  except Exception as e:
    errors.inc()  # Increment the error counter
    print(f"Error: Unable to make an API call. Detailed error: {e}")
    # Extract the first part of Estimated Effort
    effort_part1 = extract_numeric_value(response_part1.choices[0].text) if response_part1.choices else None

    # Calculate the second part of Estimated Effort (Remaining Team members * 0.5)
    remaining_team_members = row['teamsize'] - row['dedicatedteammembers']
    effort_part2_prompt = f"Calculate the second part of Estimated Effort using the formula: Remaining Team members
({remaining_team_members}) * 0.5."

    # Make an API call for the second part of Estimated Effort
    try:
        with response_time.time():
            response_part2 = openai.Completion.create(
                engine="text-davinci-003",
                prompt=effort_part2_prompt,
                temperature=0.7,
                max_tokens=50,
                n=1,
                stop=None
            )
        api_calls.inc()  # Increment the API call counter

    except Exception as e:
        errors.inc()  # Increment the error counter
        print(f"Error: Unable to make an API call. Detailed error: {e}")
```

6. To send prompts to OpenAI API to create queries to
    1. Select the first 10 rows of the table.
    2. Find the Average of the predicted object points.
    3. Find the Average of the predicted estimated effort.

```
def generate_and_execute_query(prompt):
    try:
        with response_time.time():
            response = openai.Completion.create(
                engine="text-davinci-003",
                prompt=prompt,
                temperature=0.7,
                max_tokens=50,
                n=1,
                stop=None
            )
        api_calls.inc()

    except Exception as e:
        errors.inc()
        print(f"Error: Unable to make an API call. Detailed error: {e}")


    generated_query = response.choices[0].text.strip() if response.choices else None

    if generated_query:
        print(f"Generated SQL Query: {generated_query}")
        try:
            result_data = pd.read_sql_query(generated_query, engine)
            db_queries.inc()
            print("Result:")
            print(result_data)
        except Exception as e:
            errors.inc()
            print(f"Error: Unable to execute the generated SQL query. Detailed error: {e}")
    else:
        errors.inc()
        print("Error: Unable to generate a valid SQL query.")


prompts = [

    "Create an SQL query suitable for a PostgreSQL database to retrieve the first 10 rows from a table named 'TestingData'. Ensure that the table name is enclosed in double quotes to adhere to PostgreSQL's case sensitivity. The query should select all columns from these rows.",
    "Create an SQL query suitable for a PostgreSQL database to retrieve the average value of the 'objectpointspredicted' column from a table named 'TestingData'. Ensure that the table name is enclosed in double quotes to adhere to PostgreSQL's case sensitivity.",
    "Create an SQL query suitable for a PostgreSQL database to retrieve the average value of the 'estimatedeffortpredicted' column from a table named 'TestingData'. Ensure that the table name is enclosed in double quotes to adhere to PostgreSQL's case sensitivity."

]

# Execute the function for each prompt
for prompt in prompts:
    generate_and_execute_query(prompt)
```