Question-1

a)

α= 0.05

30.87-30.68=0.19

z=1.96

0.19$\pm\ 1.96\sqrt{\left(\frac{.10^2}{12}+\frac{1.5^2}{10}\right)}$

0.19$\pm\ 1.96(.0552)$

0.19$\pm$ .10883

So, the confidence interval will be:

(.08117, .29833)

b)

$$\frac{30.87-30.68}{\sqrt{\frac{.10^2}{12}+\frac{.15^2}{10}}}$$

=3.421711

Critical Z value is 1.96

3.42 is greater than 1.9, hence, we would reject H0

c) 2 x (1- $\phi(3.42)$)

2 x(1-.996889)

=.0006

Question-2

a) 18 - 24 -1.96$\sqrt{\frac{3^2}{20}+\frac{3^2}{20}}\ \le\ u_1-u_2\le\ 18\ -\ 24$ +1.96$\sqrt{\frac{3^2}{20}+\frac{3^2}{20}}$

-7.8594 $\le\ u_1-u_2\le\ \ -\ 4.14058$

Hence, the confidence interval value is between (-7.8594, $\le\ \ -\ 4.14058$)

b) $z_0=\frac{18-24}{\sqrt{\frac{3^2}{20}+\frac{3^2}{20}}}$

= -6.325

Critical value of Z at $\alpha\ =\ 0.5$ significance level is 1.96

we will reject the null hypothesis.

Hence, we can say that the propellants had different burning rate.

c) P-value:

$2\ \times\ P(Z\ \ge\ 6.325)$

= 2 x (1-$\Phi(6.325)$)= 2 x (1-1) = 0

Question-3

In the R file.

Question -4

(a) Perform all calculations by-hand. Consider the data in following:

$$x = 0, 1, 2, 3, 4, 5$$

$$y = -0.01, 1.2, 1.97, 3.15, 4.02, 5.01$$

i. Use a linear regression analysis to fit the data using a linear function $f(x) = a_0 + a_1 x$. Report the 95% confidence interval for the slope.

ii. Now use a linear regression analysis to fit the data using a quadratic polynomial $f(x) = a_0 + a_1 x + a_2 x^2$.

iii. Which of the two functions do you think is the more appropriate fit of the data? Hint: Think about the value of a_2 for the quadratic function.

```
import math
x = np.array([0, 1, 2, 3, 4, 5])
y = np.array([-0.01, 1.2, 1.97, 3.15, 4.02, 5.01])
xmean=sum(x)/len(x)
print("xmean",xmean)
ymean=sum(y)/len(y)
print("ymean",ymean)

sum1=0
sum2=0
for i in range(6):
  sum1+=((x[i]-xmean)*(y[i]-ymean))
  sum2+=(((x[i]-xmean)**2))

print("sum1",sum1)
print("sum2",sum2)

m=sum1/((sum2))
print("m",m)

b=ymean-(m*xmean)
print("b",b)

print(f"The linear regression equaton would be: y = {m}x + {b}")
```

```
xmean 2.5
ymean 2.5566666666666666
sum1 17.369999999999997
sum2 17.5
m 0.9925714285714284
b 0.07523809523809533
The linear regression equaton would be: y = 0.9925714285714284x + 0.07523809523809533
```

i)

y= mx + b

xmean = $\frac{0+1+2+3+4+5}{6} = 2.5$

$$ymean = \frac{-.01 + 1.2 + 1.97 + 3.15 + 4.02 + 5.01}{6} = 2.556667$$

$$\sum(x - \overline{x}) * (y - \overline{y}) = 44.4672$$

$$\sum(x - \overline{x})^2 = 17.5$$

$$m = \frac{\sum(x-\overline{x}) * (y-\overline{y})}{\sum(x-\overline{x})^2} = .99257$$

$$b = ymean - m \times xmean = 2.556667 - (.99257 \times 2.5) = .075$$

$$y = .99257x + .075$$

```python
import numpy as np

x = np.array([0, 1, 2, 3, 4, 5])
y = np.array([-0.01, 1.2, 1.97, 3.15, 4.02, 5.01])
a1 = 0.9925714285714284  # Slope from your regression

# predicted values
y_hat = a1 * x + 0.07523809523809533

# mean of x
x_mean = np.mean(x)

# 3. Calculate SS_xx
SS_xx = np.sum((x - x_mean)**2)
print(SS_xx)

RSS = np.sum((y - y_hat)**2)
print(RSS)

#Calculate SE
n = len(x)
SE_a1 = np.sqrt(RSS / ((n - 2) * SS_xx))
print(SE_a1)

# 95% confidence with 4 degrees of freedom
t_stat = 2.776  #from t-distribution table

# 7. Calculate Confidence Interval
CI_lower = a1 - t_stat * SE_a1
CI_upper = a1 + t_stat * SE_a1

print(f'Confidence Interval: ({CI_lower}, {CI_upper})')
```

```
17.5
0.04376761904761909
0.025005033506881626
Confidence Interval: (0.923157455556325, 1.0619854015865318)
```

For 95% confidence interval:

$$SE = \sqrt{\frac{\sum (y-\hat{y})^2}{(n-2) \times \sum (x-\bar{x})^2}} = .025$$

Confidence lower bound= .99257 - (2.776 x .025) = .923
Confidence upper bound= .99257 + (2.776 x .025) = 1.06 CI= (.923 ,1.06)

ii)

```python
x = [0, 1, 2, 3, 4, 5]
y = [-0.01, 1.2, 1.97, 3.15, 4.02, 5.01]

# Calculate necessary sums
sum_x = sum(x)
sum_y = sum(y)
sum_x2 = sum(xi**2 for xi in x)
sum_x3 = sum(xi**3 for xi in x)
sum_x4 = sum(xi**4 for xi in x)
sum_xy = sum(xi*yi for xi, yi in zip(x, y))
sum_x2y = sum(xi**2 * yi for xi, yi in zip(x, y))

# Number of data points
n = len(x)

# Matrix for: AX = B
A = [
    [n, sum_x, sum_x2],
    [sum_x, sum_x2, sum_x3],
    [sum_x2, sum_x3, sum_x4]
]
B = [sum_y, sum_xy, sum_x2y]
print(f'sum of x {sum_x}, sume of y {sum_y}, sum of x^2 {sum_x2}, sum of x^3 {sum_x3}, sum of x4 {sum_x4}, sum of xy {sum_xy}, sum of x^2y {sum_x2y}')

print(f"Matrix A: \n",A)
print(f"Matrix B: \n",B)


# Solving the linear system A * [a0, a1, a2] = B
def solve_linear_system(A, B):
    from numpy.linalg import inv
    return list(inv(np.array(A)).dot(B))

a0, a1, a2 = solve_linear_system(A, B)

print(f"Quadratic Fit: a0 = {a0}, a1 = {a1}, a2 = {a2}")
```

```
sum of x 15, sume of y 15.34, sum of x^2 55, sum of x^3 225, sum of x4 979, sum of xy 55.72, sum of x^2y 227.0
Matrix A:
 [[6, 15, 55], [15, 55, 225], [55, 225, 979]]
Matrix B:
 [15.34, 55.72, 227.0]
Quadratic Fit: a0 = 0.033571428571412085, a1 = 1.0550714285714342, a2 = -0.012499999999999654
```

$$\sum x = 15$$

$$\sum y = 15.34$$

$$\sum x^2 = 55$$

$$\sum x^3 = 255$$

$$\sum x^4 = 979$$

$$\sum xy = 55.72$$

$$\sum x^2 y = 227$$

$$\begin{pmatrix} 6 & 15 & 55 \\ 15 & 55 & 225 \\ 55 & 225 & 979 \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 15.34 \\ 55.72 \\ 227 \end{pmatrix}$$

a_0 = .033
a_1 = 1.055
a_2 = -0.0125
iii)
Since a_2 is very small, it indicates that the quadratic term is not contributing much, and a linear model might suffice.
So linear function will be a more appropriate fit for data.

b)

```
# Given data
T = np.array([1, 20, 60, 80, 100])  # Temperature in Celsius
mu = np.array([2.5e-3, 3.5e-4, 4.98e-5, 2.12e-5, 1.05e-5])  # Kinematic viscosity in m^2 s^-1

# Transforming the data for linear regression
X = 1 / T
print("X:", X)
Y = np.log(mu)
print("Y: ",Y)

# Number of data points
N = len(T)
print("N: ", N)

# Calculating sums for the linear regression formulas
sum_X = np.sum(X)
sum_Y = np.sum(Y)
sum_XY = np.sum(X * Y)
sum_X_squared = np.sum(X**2)

print(f" sum X: {sum_X}, sum_Y: {sum_Y}, sum_XY: {sum_XY} , sum_X_squared: {sum_X_squared})")

# Calculating the slope (m) and y-intercept (C)
m = (N * sum_XY - sum_X * sum_Y) / (N * sum_X_squared - sum_X**2)
C = (sum_Y - m * sum_X) / N

print("m: ", m)
print("C: ", C)

# Calculating a and b from m and C
b = -m
a = np.exp(C)



# Print the results
print("a =", a)
print("b =", b)
```

```
X: [1.         0.05       0.01666667 0.0125     0.01      ]
Y:  [ -5.99146455  -7.9575774   -9.90749557 -10.76150938 -11.4641353 ]
N:  5
 sum X: 1.0891666666666666, sum_Y: -46.08218220160957, sum_XY: -6.803628563725837 , sum_X_squared: 1.0030340277777778)
m:  4.223952735415945
C:  -10.136554144520021
a = 3.960504121013328e-05
b = -4.223952735415945
```

T = [1, 20, 60, 80, 100]

$\mu = [2.5 \times 10^{-3}, 3.5 \times 10^{-4}, 4.98 \times 10^{-5}, 2.12 \times 10^{-5}, 1.05 \times 10^{-5}]$

$X = \frac{1}{T}$ = [1, .05, .0166, .0125, .01]

$\sum X =$ 1.089

$\sum X^2 = 1.003$

Y = log(μ) = [-5.99146455 -7.9575774 -9.90749557 -10.76150938 -11.4641353]

$\sum Y = -46.082$

$\sum XY = -6.80$

$$m = \frac{N \sum XY - \sum X \times \sum Y}{N \sum X^2 - (\sum X)^2} = \frac{(5 \times -6.80) - 1.089 \times -46.082}{5 \times 1.003 - 1.186} = 4.22$$

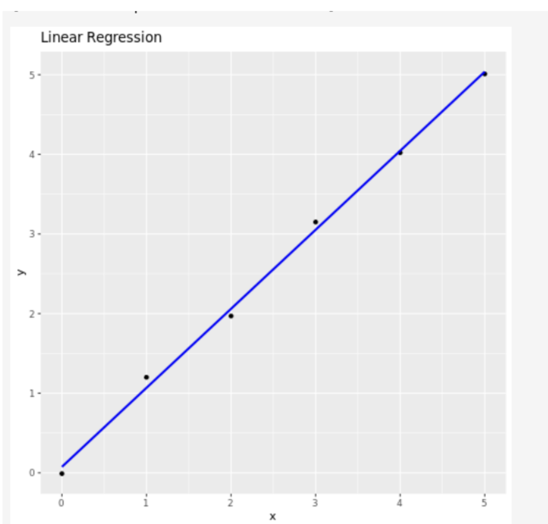
$$c = \frac{\sum Y - m \times \sum X}{5} = -10.136$$

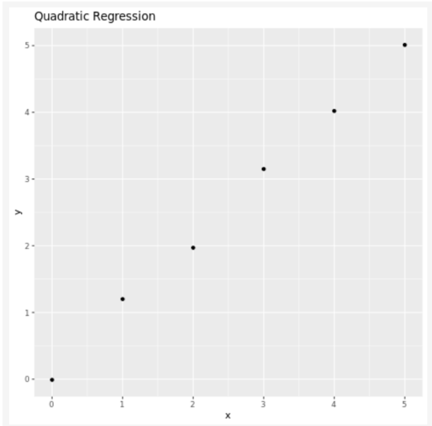
b= -m = -4.22
a=exp(c) = 3.96

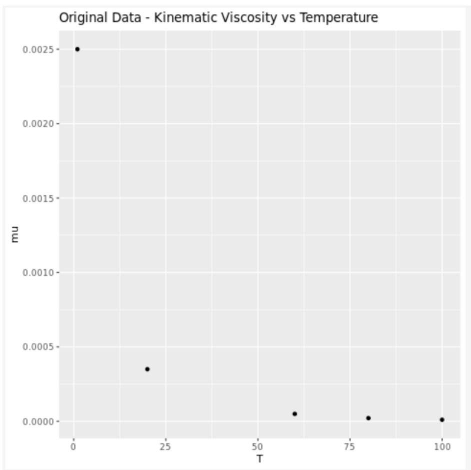
```r
# Load necessary library
library(ggplot2)

# Part 4(a)i - Linear Regression
x <- c(0, 1, 2, 3, 4, 5)
y <- c(-0.01, 1.2, 1.97, 3.15, 4.02, 5.01)

# Perform linear regression
linear_model <- lm(y ~ x)

# Compute and display confidence intervals and p-values for linear regression
cat("Linear Regression Coefficients (95% CI and p-values):\n")
print(summary(linear_model)$coefficients)
print(confint(linear_model, level = 0.95))

# Plot the data and the linear regression line
ggplot() +
  geom_point(aes(x, y)) +
  geom_smooth(method = "lm", aes(x, y), se = FALSE, color = "blue") +
  ggtitle("Linear Regression")

# Part 4(a)ii - Quadratic Regression
quadratic_model <- lm(y ~ x + I(x^2))

# Compute and display confidence intervals and p-values for quadratic regression
cat("\nQuadratic Regression Coefficients (95% CI and p-values):\n")
print(summary(quadratic_model)$coefficients)
print(confint(quadratic_model, level = 0.95))

# Plot the data and the quadratic regression curve
ggplot() +
  geom_point(aes(x, y)) +
  stat_smooth(method = "lm", formula = y ~ poly(x, 2), se = FALSE, color = "red") +
  ggtitle("Quadratic Regression")

# Part 4(b) - Exponential Model Transformation and Regression
T <- c(1, 20, 60, 80, 100)
mu <- c(2.5e-3, 3.5e-4, 4.98e-5, 2.12e-5, 1.05e-5)

# Transforming the data for linear regression
transformed_T <- 1 / T
transformed_mu <- log(mu)

# Perform linear regression on transformed data
exp_model <- lm(transformed_mu ~ transformed_T)

# Compute and display confidence intervals and p-values for exponential model
cat("\nExponential Model Regression Coefficients (95% CI and p-values):\n")
print(summary(exp_model)$coefficients)
print(confint(exp_model, level = 0.95))

# Plot the original data and transformed data
ggplot() +
  geom_point(aes(T, mu)) +
  ggtitle("Original Data - Kinematic Viscosity vs Temperature")

```

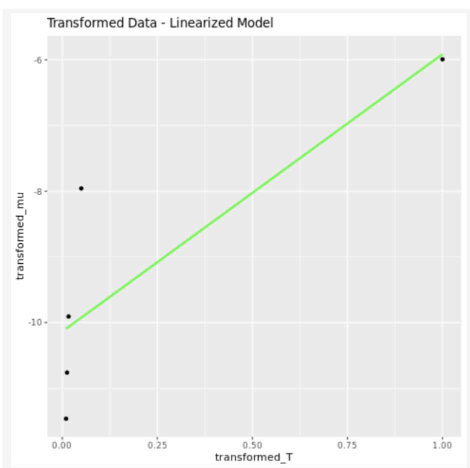
```r

ggplot() +
  geom_point(aes(transformed_T, transformed_mu)) +
  geom_smooth(method = "lm", aes(transformed_T, transformed_mu), se = FALSE, color = "green") +
  ggtitle("Transformed Data - Linearized Model")

```

```
Linear Regression Coefficients (95% CI and p-values):
             Estimate Std. Error    t value     Pr(>|t|)
(Intercept) 0.0752381 0.07570650  0.9938129 3.765655e-01
x           0.9925714 0.02500503 39.6948650 2.406460e-06
                 2.5 %   97.5 %
(Intercept) -0.1349568 0.285433
x            0.9231463 1.061997
`geom_smooth()` using formula 'y ~ x'

Quadratic Regression Coefficients (95% CI and p-values):
               Estimate Std. Error    t value    Pr(>|t|)
(Intercept)  0.03357143 0.10191550  0.3294045 0.763508092
x            1.05507143 0.09586466 11.0058438 0.001606355
I(x^2)      -0.01250000 0.01840378 -0.6792084 0.545714529
                  2.5 %     97.5 %
(Intercept) -0.29076917 0.35791203
x            0.74998731 1.36015555
I(x^2)      -0.07106903 0.04606903

Exponential Model Regression Coefficients (95% CI and p-values):
                Estimate Std. Error    t value     Pr(>|t|)
(Intercept)   -10.136554  0.7381237 -13.732866 0.0008355232
transformed_T   4.223953  1.6479966   2.563083 0.0829909964
                   2.5 %    97.5 %
(Intercept)   -12.485593 -7.787515
transformed_T  -1.020708  9.468614
`geom_smooth()` using formula 'y ~ x'

[Execution complete with exit code 0]
```




Quadratic Regression
y
x


Original Data - Kinematic Viscosity vs Temperature
mu
T


Transformed Data - Linearized Model
transformed_mu
transformed_T