PIPELINES PROJECT

STUDENTS PERFORMANCE DATA

BY NINJAS GROUP

CONTENT

STUDENTS PREFORMANE DATASET

EXPLORATORY DATA ANALYSIS (EDA)

PREPROCESSING TECHNIQUEES

FINDINGS

STUDENTS PERFORMANCE DATASET

8

Total Columns

1000+

rows

4

numeric columns

5

categorical columns

OBJECTIVE

To understand how the student's performance (test scores) is affected by the other variables (Gender, Ethnicity, Parental level of education, Lunch, Test preparation course)

STUDENT PERFORMANCE DATASET - CATEGORICAL COLUMNS

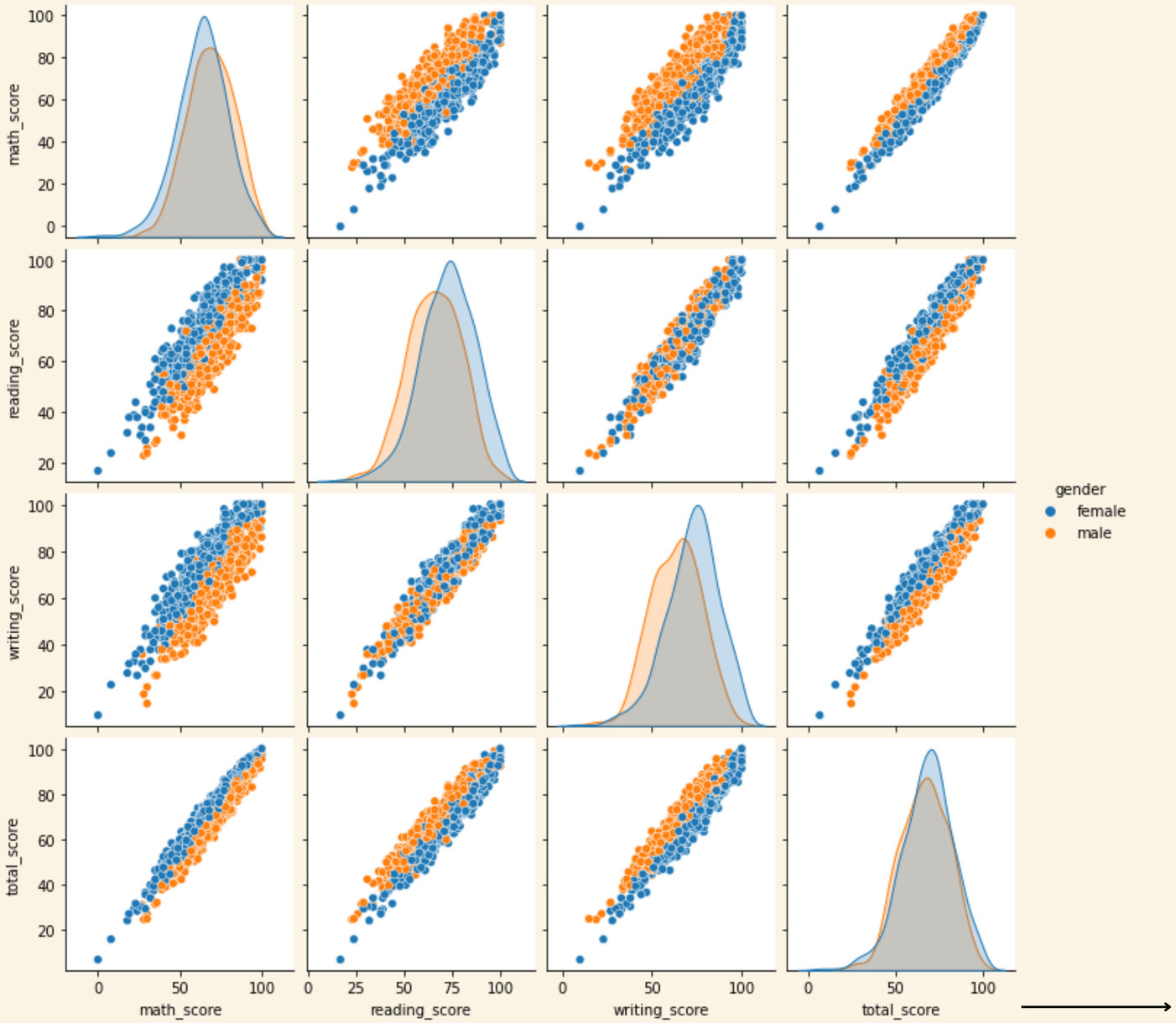
gender	race	parental_level_of_education	lunch	test_preparation_course
female	group B	bachelor's degree	standard	none
female	group C	some college	standard	complated
female	group B	master's degree	standard	none
male	group A	associate's degree	free/reduced	none
male	group C	some college	standard	none
female	group B	associate's degree	standard	none
female	group B	some college	standard	complated
male	group B	some college	free/reduced	none

STUDENT PERFORMANCE DATASET - NUMERICAL COLUMNS

math_score	reading_score	writing_score	total_score
72	72	74	72.5
69	90	88	79
90	95	93	92
47	57	44	48.75
76	78	75	76.25
71	83	78	75.75
88	95	92	90.75
40	43	39	40.5
64	64	67	64.75
38	60	50	46.5
58	54	52	55.5

EXPLORATORY DATA ANALYSIS (EDA)
PAIRPLOT

sns.pairplot(student_data, hue ='gender')



EXPLORATORY DATA ANALYSIS (EDA)
CORRELATION

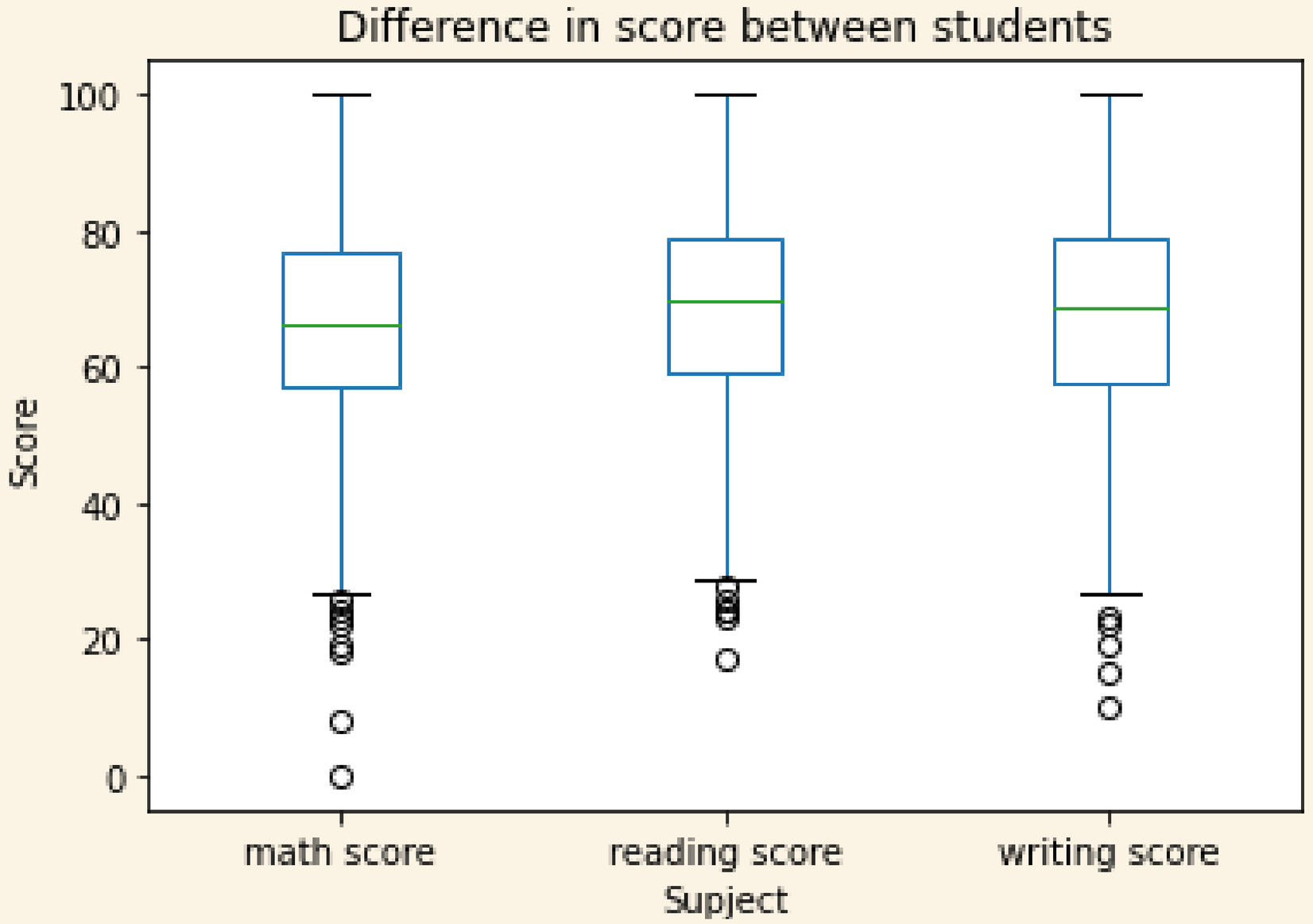
```
corr = student_data.corr()

corr.style.background_gradient(cmap='coolwarm')
```

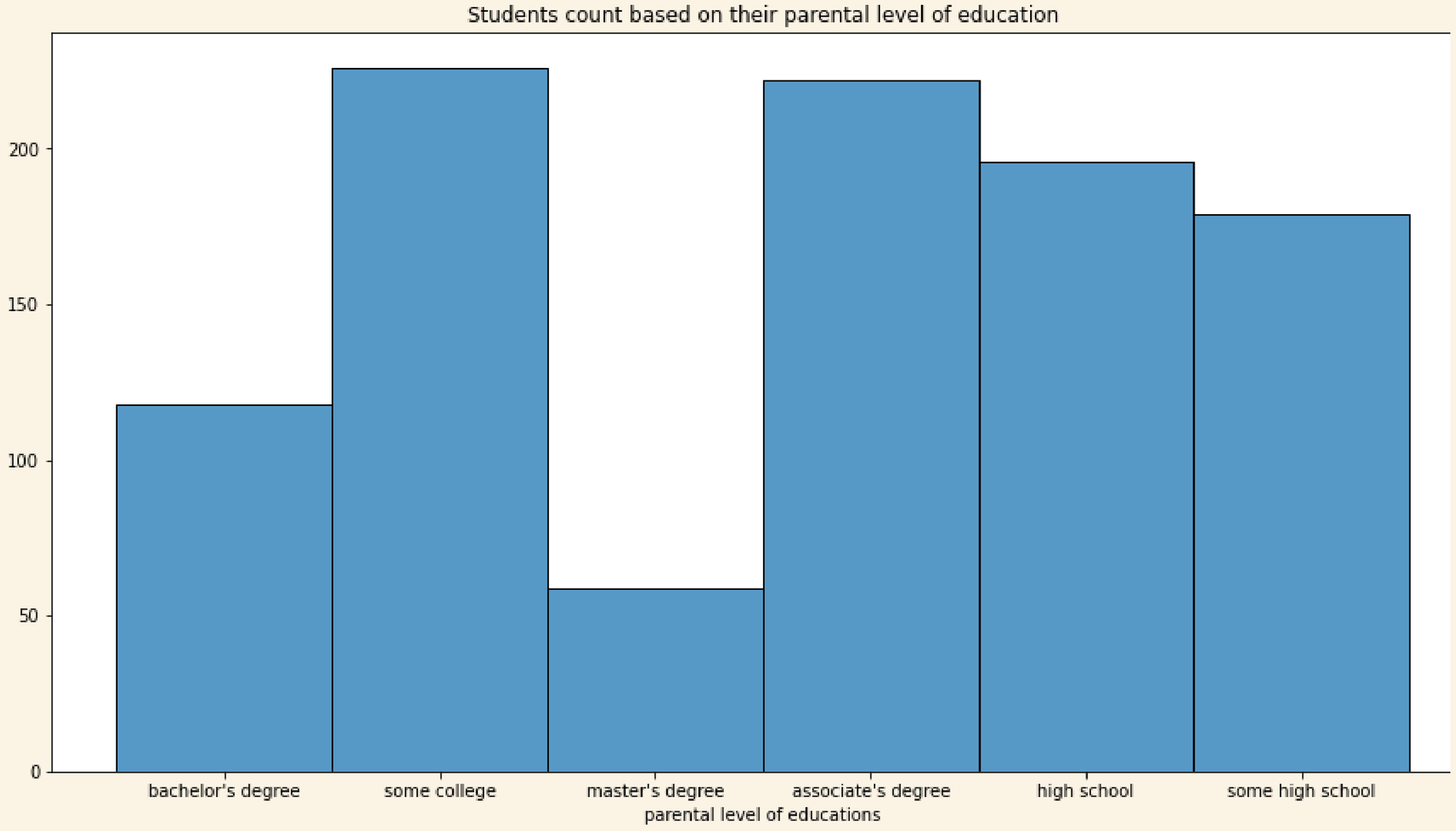
	math_score	reading_score	writing_score	total_score
math_score	1.000000	0.817580	0.802642	0.955128
reading_score	0.817580	1.000000	0.954598	0.945312
writing_score	0.802642	0.954598	1.000000	0.937841
total_score	0.955128	0.945312	0.937841	1.000000

EXPLORATORY DATA ANALYSIS (EDA)

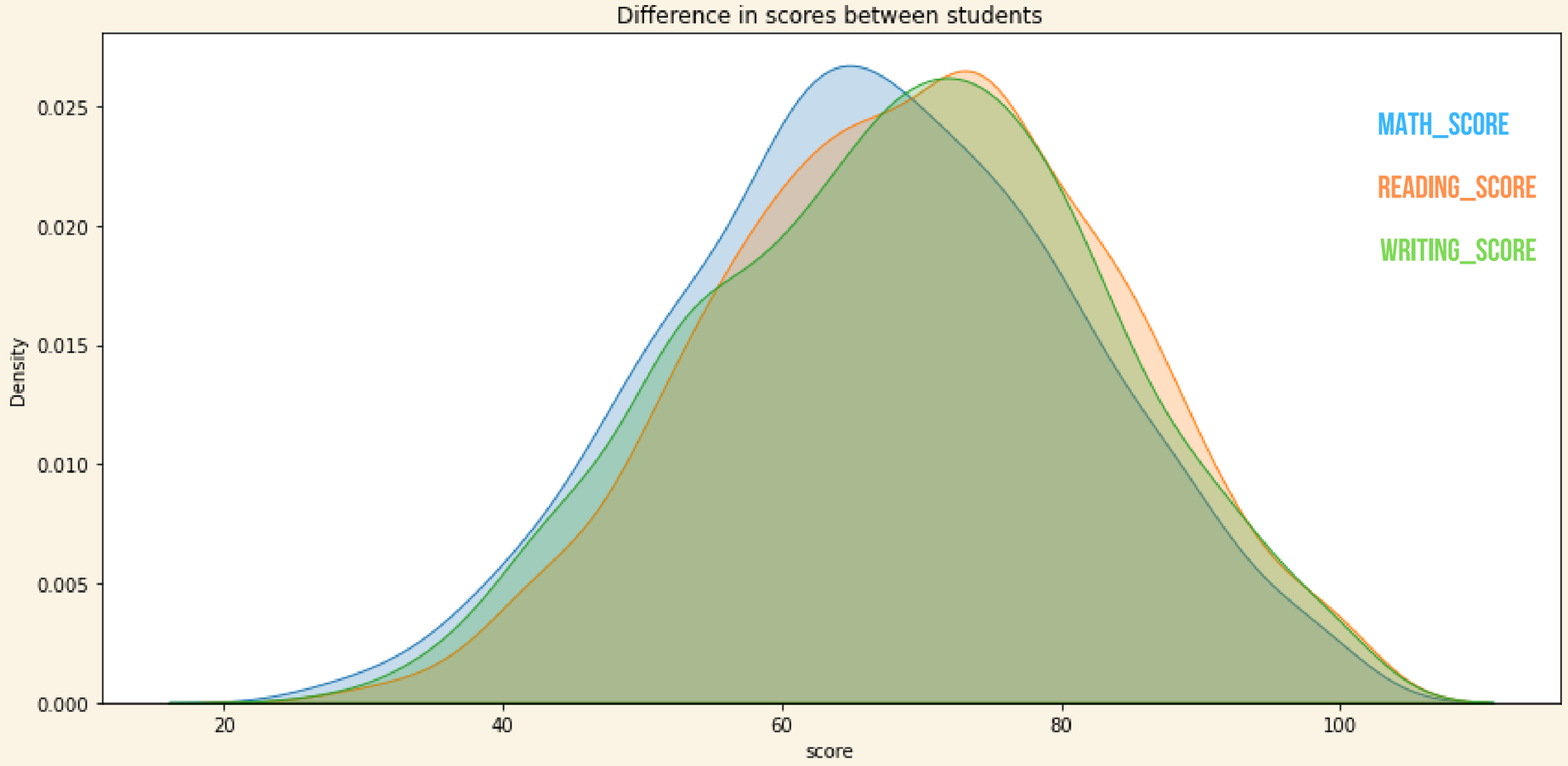
BOXPLOT



EXPLORATORY DATA ANALYSIS (EDA)
HISTOGRAM CHART



EXPLORATORY DATA ANALYSIS (EDA)
KDEPLOT



PROCESSING TECHNIQUES

SPLITTING THE DATASET

ONE HOT ENCODER

ORDINAL ENCODER

PIPLINE

DROPPING THE OUTLIERS

SPLITTING THE DATASET

```
training, test =
 train_test_split(student_data, train_size=0.8,   test_size=0.2,
random_state=42  )
```

```
test.head()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Total_score
218	male	group B	high school	free/reduced	none	66	77	70	213
337	male	group C	associate's degree	standard	none	49	51	43	143
508	male	group C	master's degree	standard	none	79	78	77	234
314	female	group C	bachelor's degree	standard	completed	59	64	75	198
91	male	group C	high school	free/reduced	none	27	34	36	97

```
training.head()
```

	gender	race/ethnicity	parental level of education	lunch	test preparation course	math score	reading score	writing score	Total_score
877	male	group C	some high school	standard	none	57	61	54	172
220	female	group D	high school	standard	completed	56	68	74	198
299	male	group D	associate's degree	free/reduced	none	90	87	75	252
50	male	group E	some college	standard	none	53	55	48	156
890	female	group E	some college	standard	completed	86	85	91	262

ONE HOT ENCODER

```
# Create our One Hot Encoder object
one_hot = OneHotEncoder()

col_names = ["gender", "lunch","test preparation course"]

# One Hot encode the column in both the train and validation sets
one_hot_df = one_hot.fit_transform(training[col_names]).toarray()

one_hot_df_val = one_hot.transform(test[col_names]).toarray()
```

```
oh_df.head()
```

	gender_female	gender_male	lunch_free/reduced	lunch_standard	test preparation course_completed	test preparation course_none
877	0.0	1.0	0.0	1.0	0.0	1.0
220	1.0	0.0	0.0	1.0	1.0	0.0
299	0.0	1.0	1.0	0.0	0.0	1.0
50	0.0	1.0	0.0	1.0	0.0	1.0
890	1.0	0.0	0.0	1.0	1.0	0.0

ORDINAL ENCODER

–

```
ordinal = OrdinalEncoder()

col_names = ["race/ethnicity","parental level of education"]

# Ordinal encode the column
ordinal_ls = ordinal.fit_transform(training[col_names])
ordinal_ls_val = ordinal.transform(test[col_names])
```

key_0	race/ethnicity	parental level of education
909	4.0	4.0
305	0.0	0.0
911	0.0	0.0
60	4.0	4.0
825	2.0	2.0

PIPLINE

```
# Only use numerical features

X_train_n = X_train.select_dtypes(exclude=["category", "object"])
X_test_n = X_val.select_dtypes(exclude=["category", "object"])

# Create a Pipeline for our model
pipe = make_pipeline(
    # 1st step handle missing values
    SimpleImputer(), # Impute missing values
    # scale columns
    StandardScaler(),
    # apply the model
    LinearRegression()
)

pipe.fit(X_train_n,y_train)
pipe.score(X_test_n, y_val)
```

DROPPING THE OUTLIERS

```
#Dropping the outliers from column
student_data=student_data.drop(student_data[student_data['math score']<27 ].index)
```

```
#Dropping the outliers from column
student_data=student_data.drop(student_data[student_data['reading score']<29 ].index)
```

```
#Dropping the outliers from column
student_data=student_data.drop(student_data[student_data['writing score']<25.25 ].index)
```

FINDINGS

key_0	parental level of education	Total_score	Actual_score	Predict_score
218	1.0	213	66	65
337	2.0	143	49	73
508	2.0	234	79	62
314	2.0	198	59	57
91	2.0	97	27	67
...	...	...	...	...
457	3.0	155	53	62
682	1.0	171	62	73
310	1.0	229	73	77
741	0.0	150	37	67
242	3.0	163	56	81

THANK YOU