MAIN REPORT

Q1:

I first convert all fits files to csv. I took this step because csv is one of the most commonly known formats while fits is a format quite specific to the Astronomy community. Hence, for a general data scientist with no knowledge of the fits format, it will be easier to deal with a csv format as it can be seen and modified with tools available in almost every operating system (while fits requires a specific fits viewer). In addition, sqlite cannot deal with data provided with fits format while it is extremely easy to deal with csv in sqlite by just putting .mode csv.

To make this pre-processing formal, I have provided the python file named fitstocsv.py. It can be modified and used by any other person in a similar situation. I also have another python file named addmjdandfieldid.py that adds two columns named 'MJD' and 'FieldID' with their values to these csv files to make another csv file of the format *-all.csv. the purpose of this will be clear when I later use these columns as foreign keys.

NOTE ON 23rd January: I missed the fact that the Ks observations are listed in order. So, I made an ordering of the different observation files based on the date and time in the filename in the File info csv file. This is exactly what you had mentioned not to do and I somehow missed that (probably because my master's project work uses the opposite convention and I extrpolated). I realized the correct order much later (today in fact, when I've done all the questions for about a week now) but correcting it now is going to take a lot of time and effort (it will affect almost all operations that follow) but right now I'm constrained by my First Year Master's Project. So, I apologize for this trangression and me being unable to correct it. My convention is as follows:

FieldID	Filename	Filter	MJD
1	Field-1-Z-all.csv	Z	57267.1671072
1	Field-1-J-all.csv	J	57257.0504323
1	Field-1-H-all.csv	H	57257.044108
1	Field-1-Ks-E001-all.csv	Ks	56788.346937
1	Field-1-Ks-E003-all.csv	Ks	56561.0020158
1	Field-1-Ks-E002-all.csv	Ks	56829.0390512
1	Field-1-Y-all.csv	Y	57267.1596647
2	Field-2-Z-all.csv	Z	57268.1671072
2	Field-2-J-all.csv	J	57258.0504323
2	Field-2-H-all.csv	H	57258.044108
2	Field-2-Ks-E001-all.csv	Ks	56789.346937
2	Field-2-Y-all.csv	Y	57268.1596647
3	Field-3-Z-all.csv	Z	57268.1671072
3	Field-3-J-all.csv	J	57258.0504323
3	Field-3-H-all.csv	H	57258.044108
3	Field-3-Ks-E001-all.csv	Ks	56789.346937
3	Field-3-Ks-E002-all.csv	Ks	56562.0020158
3	Field-3-Y-all.csv	Y	57268.1596647

These are also saved in the tabledatalocationq1.csv which forms the data for Relationtable in test2.db.

1a)

The details of the question leads to the conclusion that the database should be a 'Relational Database' because the table with the locations of the data files and the data files themselves need to be linked by foreign keys. Hence, it will be good to import all data provided to us as separate tables as well as make another table which in turn links to these data tables.

The data tables are named Stars(Field)(Filter)(Additional Number if Ks filter here named K). The table which in turn is linked to all these tables by foreign keys is named as Relationtable. The foreign keys link the Field ID and MJD of this table to the two columns named FieldID and MJD in each data table. It is important to note that here RElationtable is the child to all data tables. So, a new data can be added to the database only when the constraint laid by the foreign keys are followed. A schema for creation of this databse is provided in createtablefromcsv.sql.

Now the database test2.db is impoerted by running sqlite3 test2.db on the terminal. Then the .sql script is run as .run createtablefromcsv.sql. I have already done this, so the tables can simply be seen by typing .tables. Now, we are ready to formulate sql queries to get relevant results.

1b)

Note: Signal is taken to be Flux1 and Noise as dFlux1, Colour (Y,J,H,Ks,Z) is taken to be Mag1.

R1a:

All images observed between MJD=56800 and MJD=57300

Sql query:
```
SELECT Filename FROM Relationtable
WHERE MJD<57300.0 AND MJD>56800.0;
```

Result of query:
Field-1-Z-all.csv
Field-1-J-all.csv
Field-1-H-all.csv
Field-1-Ks-E002-all.csv
Field-1-Y-all.csv
Field-2-Z-all.csv
Field-2-J-all.csv
Field-2-H-all.csv
Field-2-Y-all.csv
Field-3-Z-all.csv
Field-3-J-all.csv
Field-3-H-all.csv
Field-3-Y-all.csv

13 out of 18 data files fulfil the criterion.

R1b:

The number of stars detected with S/N > 5 in each image

Sql query:

```
SELECT COUNT(*) as 'NStars' FROM Stars1Z
WHERE Flux1/dFlux1>5;

SELECT COUNT(*) as 'NStars' FROM Stars1J
WHERE Flux1/dFlux1>5;

SELECT COUNT(*) as 'NStars' FROM Stars1H
WHERE Flux1/dFlux1>5;

SELECT COUNT(*) as 'NStars' FROM Stars1K2
WHERE Flux1/dFlux1>5;

SELECT COUNT(*) as 'NStars' FROM Stars1Y
WHERE Flux1/dFlux1>5;

SELECT COUNT(*) as 'NStars' FROM Stars2Z
WHERE Flux1/dFlux1>5;

SELECT COUNT(*) as 'NStars' FROM Stars2J
WHERE Flux1/dFlux1>5;

SELECT COUNT(*) as 'NStars' FROM Stars2H
WHERE Flux1/dFlux1>5;

SELECT COUNT(*) as 'NStars' FROM Stars2Y
WHERE Flux1/dFlux1>5;

SELECT COUNT(*) as 'NStars' FROM Stars3Z
WHERE Flux1/dFlux1>5;

SELECT COUNT(*) as 'NStars' FROM Stars3J
WHERE Flux1/dFlux1>5;

SELECT COUNT(*) as 'NStars' FROM Stars3H
WHERE Flux1/dFlux1>5;

SELECT COUNT(*) as 'NStars' FROM Stars3Y
WHERE Flux1/dFlux1>5;
```

Result on query (in respective order):

```
9952
10000
10000
9997
9998
9992
10000
9996
10000
```

9990
10000
10000
10000

R2:

Find the objects that have J-H > 1.5

SQL query:

```
.mode csv
.output JHfield1.csv
SELECT s.StarID as n, s.Mag1-o.Mag1 as JH
FROM
        Stars1J as s
        JOIN Stars1H as o
        ON n=o.StarID
WHERE JH > 1.5;

.mode csv
.output JHfield2.csv
SELECT s.StarID as n, s.Mag1-o.Mag1 as JH
FROM
        Stars2J as s
        JOIN Stars2H as o
        ON n=o.StarID
WHERE JH > 1.5;

.mode csv
.output JHfield3.csv
SELECT s.StarID as n, s.Mag1-o.Mag1 as JH
FROM
        Stars3J as s
        JOIN Stars3H as o
        ON n=o.StarID
WHERE JH > 1.5;
```

Query Result saved as csv files (respectively) and histogram plots are drawn using the python file histogram.py:

JHfield1.csv
JHfield2.csv
JHfield3.csv

Comment on the graphs made:

As expected the graphs start from J-H = 1.5. The number of stars seem to follow an exponentially decreasing distribution after that. Also, Field 1 seems to have a overall larger number of stars overall while Field 2 seems to have the overall least for this criterion.

R3:

Find the objects where Ks differs by more than 20 times the flux uncertainty from the mean flux.

Assumption:

This is a bit vague. The way I understand it, this may mean finding the difference between Mag1 and Flux1 and checking if this exceeds twenty times the value of dFlux1. Since, the LHS isn't an absolute value, it could either be Mag1-Flux1 or Flux1-Mag1. The query for the former is provided below.

SQL query:

```
SELECT STARID as n, Mag1 as K FROM Stars1K1
WHERE (Mag1-Flux1)>(20.0*dFlux1);

SELECT STARID as n, Mag1 as K FROM Stars1K2
WHERE (Mag1-Flux1)>(20.0*dFlux1);

SELECT STARID as n, Mag1 as K FROM Stars1K3
WHERE (Mag1-Flux1)>(20.0*dFlux1);

SELECT STARID as n, Mag1 as K FROM Stars2K1
WHERE (Mag1-Flux1)>(20.0*dFlux1);

SELECT STARID as n, Mag1 as K FROM Stars3K1
WHERE (Mag1-Flux1)>(20.0*dFlux1);

SELECT STARID as n, Mag1 as K FROM Stars3K2
WHERE (Mag1-Flux1)>(20.0*dFlux1);
```

Result of Respective queries:

```
None
None
None
None
None
301288|15.0318
```

For the later case,

SQL query:

```
.mode csv
.output FMdF1K1.csv
SELECT STARID as n, Mag1 as K FROM Stars1K1
WHERE (Flux1-Mag1)>(20.0*dFlux1);

.mode csv
.output FMdF1K2.csv
SELECT STARID as n, Mag1 as K FROM Stars1K2
WHERE (Flux1-Mag1)>(20.0*dFlux1);
```

```
.mode csv
.output FMdF1K3.csv
SELECT STARID as n, Mag1 as K FROM Stars1K3
WHERE (Flux1-Mag1)>(20.0*dFlux1);

.mode csv
.output FMdF2K1.csv
SELECT STARID as n, Mag1 as K FROM Stars2K1
WHERE (Flux1-Mag1)>(20.0*dFlux1);

.mode csv
.output FMdF3K1.csv
SELECT STARID as n, Mag1 as K FROM Stars3K1
WHERE (Flux1-Mag1)>(20.0*dFlux1);

.mode csv
.output FMdF3K2.csv
SELECT STARID as n, Mag1 as K FROM Stars3K2
WHERE (Flux1-Mag1)>(20.0*dFlux1);
```

Results of queries are stored in the respective csv files mentioned as .output. The data from these files in made into histograms with the help of TOPCAT (bandwidth = 0.1) with format (FieldID)K(Observation number).png.

Comments on the images:
It is clear that different observations in the same field give different numbers of stars overall. 1K1 has the least amount of stars and 1K3 has the most in Field 1. Even the shape of the distribution in 1K3 is different (bimodal) than the rest two in Field 1. The difference in distribution is reduced in Field 3, however FK2 has more stars than FK1. In general, the distributions of all stars in different fields tends to be the same.

R4:

Find all catalogues that exist for a given field.

SQL query:

```
SELECT Filename FROM Relationtable
WHERE FieldID = 1;

SELECT Filename FROM Relationtable
WHERE FieldID = 2;

SELECT Filename FROM Relationtable
WHERE FieldID = 3;
```

Result of queries:
Field-1-Z-all.csv
Field-1-J-all.csv
Field-1-H-all.csv

Field-1-Ks-E001-all.csv
Field-1-Ks-E003-all.csv
Field-1-Ks-E002-all.csv
Field-1-Y-all.csv

Field-2-Z-all.csv
Field-2-J-all.csv
Field-2-H-all.csv
Field-2-Ks-E001-all.csv
Field-2-Y-all.csv

Field-3-Z-all.csv
Field-3-J-all.csv
Field-3-H-all.csv
Field-3-Ks-E001-all.csv
Field-3-Ks-E002-all.csv
Field-3-Y-all.csv

R5:

For a given field I would like to retrieve the Y, Z, J, H and Ks magnitudes for all stars with S/N > 30 in Y, Z, J, H and Ks. (Done for all fields individually)

SQL query:

```
.mode csv
.output allcoloursfield1.csv
SELECT a.starID as n, a.Mag1 as Y, b.Mag1 as Z, c.Mag1 as H, d.Mag1 as J, e.Mag1 as K1, f.Mag1 as K2, g.Mag1 as K3
FROM
        Stars1Y as a
        JOIN STars1Z as b ON  n=b.StarID
        JOIN Stars1H as c ON  n=c.StarID
        JOIN Stars1J as d ON  n=d.StarID
        JOIN Stars1K1 as e ON  n=e.StarID
        JOIN Stars1K2 as f ON  n=f.StarID
        JOIN Stars1K3 as g ON  n=g.StarID
WHERE a.Flux1/a.dFlux1 > 30 AND b.Flux1/b.dFlux1 > 30 AND c.Flux1/c.dFlux1 > 30 AND d.Flux1/d.dFlux1 > 30 AND e.Flux1/e.dFlux1 > 30 AND f.Flux1/f.dFlux1 >30 AND g.Flux1/g.dFlux1 > 30;

.mode csv
.output allcoloursfield2.csv
SELECT a.starID as n, a.Mag1 as Y, b.Mag1 as Z, c.Mag1 as H, d.Mag1 as J, e.Mag1 as K1
FROM
        Stars2Y as a
        JOIN STars2Z as b ON  n=b.StarID
        JOIN Stars2H as c ON  n=c.StarID
        JOIN Stars2J as d ON  n=d.StarID
        JOIN Stars2K1 as e ON  n=e.StarID
WHERE a.Flux1/a.dFlux1 > 30 AND b.Flux1/b.dFlux1 > 30 AND c.Flux1/c.dFlux1 > 30 AND
```

```
d.Flux1/d.dFlux1 > 30 AND e.Flux1/e.dFlux1 > 30;

.mode csv
.output allcoloursfield3.csv
SELECT a.starID as n, a.Mag1 as Y, b.Mag1 as Z, c.Mag1 as H, d.Mag1 as J, e.Mag1 as K1, f.Mag1 as K2
FROM
        Stars3Y as a
        JOIN STars3Z as b ON  n=b.StarID
        JOIN Stars3H as c ON  n=c.StarID
        JOIN Stars3J as d ON  n=d.StarID
        JOIN Stars3K1 as e ON  n=e.StarID
        JOIN Stars3K2 as f ON  n=f.StarID
WHERE a.Flux1/a.dFlux1 > 30 AND b.Flux1/b.dFlux1 > 30 AND c.Flux1/c.dFlux1 > 30 AND d.Flux1/d.dFlux1 > 30 AND e.Flux1/e.dFlux1 > 30 AND f.Flux1/f.dFlux1 >30;
```

Results of queries are stored in the .csv files named in the queries. Histograms for each colour for each field is extracted using TOPCAT (bandwidth = 0.1) and saved as Field(FieldID)(Colour)(Observation number for K Colour).png.

1c)

Constructing a table of Stars with Y-J and J-H magnitudes. Thwn using this table to create a simulated list of 100,000 stars for the Euclid mission.

Assumption:
To do this exercise, I assume that the distribution of stars I get from the database belong to a sample that can be reconstructed from a Kernel Density Estimation to construct a probability density distribution. That would mean assuming these stars are a representative sample of the distribution. Also, I assume a minimum S/N ratio of 5 to assume good quality observations.

SQL query:

```
.mode csv
.headers on
.output YJHcatalog.csv

SELECT s.StarID as n, s.Mag1-o.Mag1 as YJ, o.Mag1-p.Mag1 as JH
FROM Stars1Y as s
        JOIN Stars1J as o ON n=o.StarID
        JOIN Stars1H as p ON o.StarID = p.StarID
WHERE s.Flux1/s.dFlux1 > 5 AND o.Flux1/o.dFlux1 > 5 AND p.Flux1/p.dFlux1 > 5
UNION
SELECT s.StarID as n, s.Mag1-o.Mag1 as YJ, o.Mag1-p.Mag1 as JH
FROM Stars2Y as s
        JOIN Stars2J as o ON n=o.StarID
        JOIN Stars2H as p ON o.StarID = p.StarID
WHERE s.Flux1/s.dFlux1 > 5 AND o.Flux1/o.dFlux1 > 5 AND p.Flux1/p.dFlux1 > 5
UNION
SELECT s.StarID as n, s.Mag1-o.Mag1 as YJ, o.Mag1-p.Mag1 as JH
```

```
FROM Stars3Y as s
        JOIN Stars3J as o ON n=o.StarID
        JOIN Stars3H as p ON o.StarID = p.StarID
WHERE s.Flux1/s.dFlux1 > 5 AND o.Flux1/o.dFlux1 > 5 AND p.Flux1/p.dFlux1 > 5;
```

Now the output catalog is used as an input to the Gaussian Kernel Density estimation done in q1_c_simulation.ipynb to create a simulated list of 100,000 stars. I use Gaussian Kernel because at the moment (sklearn restriction) random samples can be drawn from the distribution constructed using Gaussian and Tophat kernels. Gaussian kernel is in general more preferable for a large distribution like the one we're using.

REFERENCES FOR Q1:

http://www.sqlite.org/foreignkeys.html
https://www.sqlservercentral.com/Forums/Topic1540629-3077-1.aspx
http://www.sqlitetutorial.net/sqlite-export-csv/
http://www.dofactory.com/sql/join
Lecture note 1 and codes used in assignments

Q2:

All questions have been answered in the q2.ipynb notebook. That should sreve as a more comprehensive report. But, I'm reproducing some part of the answer here to make this report complete in a sense. But, please read that notebook side by side for a better experience.

2a)

To understand Feature Extraction, it is imperative that we understand what 'feature' means. As I understand it, feature is analogues to 'variable' used in Physics. Thus, a feature is a set of input variables that can be measured independently. In fact, measurement of variables is essential to predict some outcome based on relationships between them. However, there can also be an extention to this definition. New features can be constructed from a combination of other pre-existing features using various mathematical tools.

In this sense, we can now begin to define feature extraction. In this context, we have been given a set of data where one column namely the redshift is thought of to be a combination of four other columns which are colours that can be measured. Hence, we term the four colour combinations as features. But, it might turn out that redshift might not depend on combination of all these features i.e some features might be less important or might be dependant on other features or redshift might not be directly related to these features, but in turn is related to an operation done on those features i.e newly constructed features. This entire process of figuring out the 'true variables' is called as Feature Extraction.

Note: This is fundamentally different from Feature Selection because Feature Selection assumes that we already know a list of features and hence need to know the best of them, while Feature Extraction doesn't assume anything.

2b+c)

I first use pandas to extract the data from the csv files. I use csv format again because of the reasons I stated in Q1. In addition to that, pandas is a pretty powerful method of dealing with tables, hence the conversion to csv from vot was also influenced by this fact.

Now, I construct a linear estimator. I do not use the entire Train data in File A for fitting. Getting a linear algorithm using the entire train dataset will cause an overfit of the data by the algorithm to a very strong extent. Hence, instead we need to train this dataset by using chunks of the first dataset to first train the algorithm and then try to fit it with another chunk of the same dataset to reduce the overall overfitting. However, the model still remains biased to this dataset.

For tuning the model, we now divide the train data into sets of 'train' and 'test' datasets using k-fold cross validation. Generally, the number of foldings (k) is either 5 or 10. This is a large dataset with about 70000 entries. Hence, I use k=5 to save time. The code is inspired from the code used in the assignments. Then we use a Linear Regression model to fit the train data sets and predict values for the test data sets. This then gives us the 'Training Error' for the first data set. However, this is not the true error. Since, we use exclusively one dataset to train our algorithm, the algorithmn might be biased towards fitting the first set of data only. Hence, a second set of data taken at a different time or place can say us how well this model actually works. This is done by applying the regressor to Xgen from the second dataset and comparing it with Ygen. This error is called 'Generalization Error'. The training error is shown first followed by the generalization error which is obatained by testing the trained model on the second dataset.

The training error lies in the range of (0.0144, 0.0166). The generalization error lies in the range of (0.0151, 0.0154). It is easily seen that the train and test error are comparable and the parameters for the model are not large (see the values of the printed coefficients and intercept in the notebook). Thus, we don't need a regularization technique like ridge or lasso. It also means the training has been done well and the data actually follows a linear trend (R^2 value is about 0.8).

2d)

Now that we have tried out a Linear Estimator, let's use a non-linear estimator of our choice. I use k-Nearest Neighbours regressor. I personally favour this because I understand it more clearly among all non-linear methods. However, there is also another reason to use this. The features we consider have only 4 dimensions which isn't much and all features can very well be assumed to be continuous. KNN is very effective in these situations.

For KNN I take a high number of neighbours (10 in number) for a good averaging and I weigh all of them by distance, so that the closer points get more prefrence. This is partly inspired by the fact that I already know that the data trend is fairly linear but it still has a significant amount of variation from a linear trend. This is known from seeing the linear scores (l.score(X,Y)) computed in the linear model iterations with almost all of them being close to a value of 0.8.

I again train the data instead of fitting it all. The need for this will be clear in the disadvantages section of this method. The training errors are in the range (0.012, 0.014) and the generalization errors are in the range (0.0127, 0.013). Hence, the estimator generalizes remarkably well. Both errors are also significantly lesser than their counterparts for the linear estimator.

2e)

Advantages of KNN over Linear:

1. KNN estimator is very flexible as it is non-parametric. So, almost all values in the training set are considered for fitting. This can also be a disadvantage as we will see later.
2. There are only 4 features and the data is fairly linear, so careful use of the non-parametric feature will better constrain the model while training. In fact, training by KNN is actually more effective here. If we would have fit the entire train file without weighting and with less neighbours, the training error will be very low (See Appendix of the notebook where a simple fitting gives training error 0 but significant generalization error). But, it will overfit the data to a very large amount, so the generalization error is larger.

Disadvantages:

1. The flexibilty of the approach also means that it is prone to interference from outlier data. This would mean that we would be in a sense overfitting the data if we don't take considerable care in selecting the number of neighbours and the weighting.
2. The KNN estimator is also computationally more intensive. It searches for neighbours and gives weights by calculating the distance. This takes longer the more features there are. Fortunately, we have only 4 features.

REFERENCES FOR Q2:

The material uses the scikit-learn software package version 0.19.1. I also took help from the online documentation of the packages used to write my code. In addition, code from the assignment was also liberally used.

ADDITIONAL REFERENCES:

Any additional references i.e websites and stack overflow sites not listed here are listed in the python files.