Sorting and Counting - Exercises 1

1. Find top 10 files by size in your home directory including the subdirectories. Sort them by size and print the result including the size and the name of the file (hint: use find with -size and -exec ls -s parameters)
2. Create a dummy file with this command : seq 15> 20lines.txt; seq 9 1 20 >> 20lines.txt; echo"20\n20" >> 20lines.txt; (check the content of file first)
   a) Sort the lines of file based on alphanumeric characters
   b) Sort the lines of file based on numeric values and eliminate the duplicates
   c) Print all duplicated lines of the file
   d) Print the line which has most repetitions
   e) Print all lines with the number of repetitions sorted by the number of repetitions from lowest to highest
3. Create another file with this command : seq 0 2 40 > 20lines2.txt
   a) Create 3rd file from the first two but without duplicates
   b) Merge the first two files. Print unique lines together with the number of occurrences inside the merged file and sorted based on line content.
4. Go to ~/Data/opentraveldata. Get the line with the highest number of engines using sort.

Sorting and Counting - Exercises 1

Find top 10 files by size in your home directory including the subdirectories. Sort them by size and print the result including the size and the name of the file (hint: use find with -size and -exec ls -s parameters)

```
find ~ -type f -size +10M -exec ls -sh {} \; | sort -nr | head
```

<path> is home the shortcut is ~ (to check home directory use: echo “$HOME”)

-size +10M

- + greater than
- - Less than
- M (megabytes)

-exec <command> {} \;

Run the command <cmd> con every match found

Then sort and fetch top 10

Sorting and Counting - Exercises 1

Create another file with this command : seq 0 2 40 > 20lines2.txt

a) Create 3rd file from the first two but without duplicates

Sort can apply sorting to several files sort <file1.txt> <file2.txt>, it just append the files and then applies sorting. Use redirect to create de new file.

a) Merge the first two files. Print unique lines together with the number of occurrences inside the merged file and sorted based on line content.

To merge files use again sort <file1.txt> <file2.txt>, instead of sorting by duplicity count, sort by line content (use sort with delimiter and column selector)

Sorting and Counting - Exercises 1

- Go to ~/Data/opentraveldata Get the line with the highest number of engines from optd_aircraft.csv by using sort. (nb_engines)

In this file each line is an aircraft, then just sort by the column nb_engines (use delimiter and column selector, remember to sort only by one column and not by a partial line)


```
A5F^Antonov^An-225^A5F^6J^A225^6^J
141^BAe^146-100^146^4J^B461^4^J
142^BAe^BAE Systems 146-200 Passenger^146^4J^B462^4^J
```

Sorting and Counting - Exercises 1

1) find ~ -type f -size +10M -exec ls -sh {} \; | sort -nr | head

2a) sort -d 20lines.txt

2b) sort -nu 20lines.txt

2c) sort -n 20lines.txt | uniq –d

2d) sort -n 20lines.txt | uniq -d -c | sort -nr | head -1

2e) sort -n 20lines.txt | uniq -c | sort -n

3a) sort -nu 20lines.txt 20lines2.txt > 20lines_no_dupl.txt

3b) sort 20lines2.txt 20lines.txt | uniq -c | sort –t “ “ -k 2n,2

4) sort -t "^" -k 7nr,7 optd_aircraft.csv |head -1

Processing and filtering - Exercises 2

Go to ~/Data/opentraveldata

1. Change the delimiter of optd_aircraft.csv to “,”
2. Check if optd_por_public.csv has repeated white spaces

   Remove blank spaces by “squeezing” with tr and use wc to measure size and compare with raw file size

```
458     1068    18612
```

```
458  1068 18612 optd_aircraft.csv
```

3. How many columns has optd_por_public.csv? (hint: use head and tr)

   Get first row with head, pipe it to tr to substitute the separator by \n then use wc to get the result

4. Print column names of optd_por_public.csv together with their column number. (hint: use paste)

   Extend former exercise by pasting seq 1 N with the previous result. Remember the operator paste <() <()

Processing and filtering - Exercises 2

Go to ~/Data/opentraveldata

Use optd_airlines.csv to obtain the airline (col name) with the most flights (col flt_freq)?

Get the position of the needed columns (name and flt_freq) and use cut to select them, then just sort properly

Use optd_airlines.csv to obtain number of Airlines (col name) in each Alliance (col alliance_code)

Airlines are unique within each alliance. Some rows have the alliance column in blank. Select the column alliance_code and count how many times they are repeated

Processing and filtering - Exercises 2

1) cat optd_aircraft.csv | tr "^“ "," | optd_aircraft_comma.csv

2) cat optd_por_public.csv | tr -s "[:blank:]" | wc
   wc optd_por_public.csv
   Compare the size in bytes!
cat optd_por_public.csv | tr –cd “ “ | wc

3) head -n 1 optd_por_public.csv| tr "^" "\n" | wc –l
cat optd_por_public.csv | tr –cd “^” | wc -c
45 delimiters=> 46 columns

4) paste <(seq 46) <(head -1 optd_por_public.csv | tr "^" "\n")

5) cat optd_airlines.csv | cut -d "^" -f 8,14 | sort -t "^" -k 2nr,2 |head -1

6) cat optd_airlines.csv| cut -d "^" -f 10 | sort| uniq -c | sort -rn | head

Processing and filtering - Exercises 3

Go to ~/Data/opentraveldata

1. Use grep to extract all 7x7 (where x can be any number) airplane models from optd_aircraft.csv.
2. Use grep to extract all 3xx (where x can be any number) airplane models from optd_aircraft.csv.

   Select the proper column (model) and then apply grep with a regular expression, remember the following key:

   [0-9]: any number, one occurrence

   [0-9]{n}: any number n occurrences

3. Use grep to obtain the number of airlines with prefix "aero" (case insensitive) in their name from optd_airlines.csv

   Same strategy, just take into account that "^any_pattern" will look for 1 occurrence of the literal "any_pattern"

4. How many optd_por_public.csv columns have "name" as part of their name? What are their numerical positions? (hint: use seq and paste)
5. Find all files with txt extension inside home directory (including all sub directories) that have **word** "Science" (case insensitive) inside the content. Print file path and the line containing the (S/s)cience word.

Processing and filtering - Exercises 3

Go to ~/Data/opentraveldata

1. Use grep to extract all 7x7 (where x can be any number) airplane models from optd_aircraft.csv.
2. Use grep to extract all 3xx (where x can be any number) airplane models from optd_aircraft.csv.

   Select the proper column (model) and then apply grep with a regular expression, remember the following key:

   [0-9]: any number, one occurrence

   [0-9]{n}: any number n occurrences

3. Use grep to obtain the number of airlines with prefix “aero” (case insensitive) in their name from optd_airlines.csv

   Same strategy, just take into account that “^any_pattern” will look for 1 occurrence of the literal “any_pattern”

4. How many optd_por_public.csv columns have “name” as part of their name? What are their numerical positions? (hint: use seq and paste)

   Remember paste <() <() and then pipe to a grep

5. Find all files with txt extension inside home directory (including all sub directories) that have **word** “Science” (case insensitive) inside the content. Print file path and the line containing the (S/s)cience word.

   Use find with the –iname to get all files with a given extension (remember wildcards!) then use –exec and grep (with case sensitive, returning the file path and the line)

Processing and filtering - Exercises 3

1) cut -d "^" -f 3 optd_aircraft.csv | grep -E "7[0-9]7"

2) cut -d "^" -f 3 optd_aircraft.csv | grep -E "3[0-9]{2}"

3) cat optd_airlines.csv | cut -d "^" -f 8 | grep -i -E "^aero" | wc –l

4) paste <(seq 50) <(head -n 1 optd_por_public.csv | tr "^" "\n") | grep name

5) find ~ -type f -iname "*.txt" – maxdepth 4 -exec grep -iwH "Science" {} \;

Processing and filtering - Exercises 4

Use Text_example.txt

1. Replace every “line” with new line character (“\n”)

   Just apply sed s/old/new/ will substitute the first occurrence of old by new, “s” is just syntax and “/” is the separator. Apply the proper option

2. Delete lines that contain the “line” word.

   sed s/old/ will remove the first occurrence of old

3. Print ONLY the lines that DON’T contain the “line” word

   To modify the default print rule, use the –n flag

   In addition to –n, “p” prints only matching lines, and “!” can reverse this behavoir

Processing and filtering - Exercises 4

1) sed 's/line/\n/g' Text_example.txt

2) sed '/line/d' Text_example.txt

3) sed -n '/line/!p' Text_example.txt

Working with compressed Files – Exercises 5

1. Go to ~/Data/us_dot/otp. Show the content of one of the files.
2. Use head/tail together with zcat command. Any difference in time execution?
3. Compress “optd_por_public.csv” with bzip2 and then extract from the compressed file all the lines starting with MAD (hint: use bzcat and grep)
4. (On_Time_On_Time_Performance_2015_1.zip): What are the column numbers of columns having “carrier” in the name ? (don't count!) (hint: we have seen this ☺)
5. (On_Time_On_Time_Performance_2015_1.zip) Print to screen, one field per line, the header and first line of the T100 file, side by side.

Working with compressed Files – Exercises 5

1. zless On_Time_On_Time_Performance_2015_1.zip
2. zcat On_Time_On_Time_Performance_2015_1.zip |head

   zcat On_Time_On_Time_Performance_2015_1.zip |tail
3. bzip2 optd_por_public.csv

   bzcat optd_por_public.csv.bz2 | grep -E "^MAD"

   or

   bzgrep -E "^MAD" optd_por_public.csv.bz2

4. paste <(seq 110) <(zcat ./On_Time_On_Time_Performance_2015_1.zip | head -n 1 | tr "," "\n")|grep -i "carrier"
5. paste <(seq 110) <(zcat ./On_Time_On_Time_Performance_2015_1.zip | head -n 1 | tr "," "\n") <(zcat ./On_Time_On_Time_Performance_2015_1.zip | head -n 2 | tail -1 | tr "," "\n")

Shell Script – Exercises 6

1. Create a script that will return column names together with their column number from the csv files. The first argument should be file name and the second delimiter.(use it on ~/Data/opentraveldata/optd_aircraft.csv)

   Get the number of columns, substituting the separator by \n and counting lines

   Paste column number and name

2. Create a script that accepts a CSV filename as input ($1 inside your script) and returns the model of the aircraft with the highest number of engines. (use it on ~/Data/opentraveldata/optd_aircraft.csv)

   Sort number of engines, then select the proper column

3. Repeat script 2, but add a second argument to accept number of a column with the number of engines. If several planes have the highest number of engines, then the script will only show one of them. (use it on ~/Data/opentraveldata/optd_aircraft.csv)

4. Create a script that accepts as input arguments the name of the CSV file, and a number (number of engines) and returns number of aircrafts that have that number of engines. (use it on ~/Data/opentraveldata/optd_aircraft.csv)

   Select the proper column

   Filter rows where the number of engines is the requested

   Count duplicates, and prepare properly the ouput (remove unnecesary blanks and select the last column from the ouptut, watch out if there is any leading blank)

Shell Script – Exercises 6

1. Create a script that will return column names together with their column number from the csv files. The first argument should be file name and the second delimiter.(use it on ~/Data/opentraveldata/optd_aircraft.csv)

```
[dsc@vm:~/Data/opentraveldata] [base] % ./column_name_numbers.sh optd_aircraft.csv "^"
1       iata_code
2       manufacturer
3       model
4       iata_group
5       iata_category
6       icao_code
7       nb_engines
8       aircraft_type
```

2. Create a script that accepts a CSV filename as input ($1 inside your script) and returns the model of the aircraft with the highest number of engines. (use it on ~/Data/opentraveldata/optd_aircraft.csv)

```
[dsc@vm:~/Data/opentraveldata] [base] 2 % ./model_with_most_engines.sh optd_aircraft.csv
The model is  An-225
```

Shell Script – Exercises 6

1. Create a script that accepts as input arguments the name of the CSV file, and a number (number of engines) and returns number of aircrafts that have that number of engines. (use it on ~/Data/opentraveldata/optd_aircraft.csv)

```
[dsc@vm:~/Data/opentraveldata] [base] % ./num_of_engines.sh optd_aircraft.csv 4
61
[dsc@vm:~/Data/opentraveldata] [base] % ./num_of_engines.sh optd_aircraft.csv 6
1
```

Shell Script – Exercises 6

To create the shebang, if /usr/bin/bash is not the appropriate path, just check where it is

```
[dsc@vm:~/Data/opentraveldata] [base] % where bash
/bin/bash
```

```
[dsc@vm:~/Data/opentraveldata] [base] % where python
/home/dsc/anaconda3/bin/python
/usr/bin/python
```

Remember to chmod your script in order to being able to execute (rwx to user)

```
[dsc@vm:~/Data/opentraveldata] [base] % ls -l
total 7,9M
-rwxrw-r-x 1 dsc dsc  187 dic 16 21:18 column_name_numbers.sh*
-rw-rw-r-- 1 dsc dsc  19K abr  2  2018 optd_aircraft.csv
-rw-rw-r-- 1 dsc dsc 105K abr  2  2018 optd_airlines.csv
-rw-rw-r-- 1 dsc dsc 7,8M abr  2  2018 optd_por_public.csv
-rw-rw-r-- 1 dsc dsc 5,7K abr  2  2018 ref_airline_nb_of_flights.csv
[dsc@vm:~/Data/opentraveldata] [base] % chmod 765 column_name_numbers.sh
```

Shell Script Exercises

./column_name_number.sh <FILE INPUT> <“DELIMITER”>

1) File: column_name_number.sh

Create the .sh file with nano or from ubutu desktop

```
#!/usr/bin/bash
FILE_INPUT=$1
DELIMITER=$2
#echo "My name is ${0}"
#echo "Delimiter= ${DELIMITER}"
#echo "file=${FILE_INPUT}"

NUM_COLUMNS=$(cat ${FILE_INPUT} | head –n 1 | tr ${DELIMITER} "\n" | wc -l)
#echo "Column Number=${NUM_COLUMNS}"

paste <(seq ${NUM_COLUMNS}) <(head –n 1 ${FILE_INPUT} | tr ${DELIMITER} "\n")
```

Get the number of columns, substituting the separator by \n and counting lines

Paste column number and name

Shell Script Exercises

./ model_with_most_engines sh <FILE INPUT>

Create the .sh file with nano
or from ubutu desktop

2) File: model_with_most_engines.sh

```
#!/usr/bin/bash
FILE_INPUT=$1

MODEL=$(sort -t "^" -k 7nr ${FILE_INPUT} | head –n 1 | cut -d "^" -f 3)
echo "The model is ${MODEL}"
```

Sort number of engines, then
select the proper column

Shell Script Exercises

3) File: model_with_most_engines2.sh

```
#!/usr/bin/bash
FILE_INPUT=$1
COLUMN_INPUT=$2


MODEL=$(sort -t "^" -k ${COLUMN_INPUT}nr ${FILE_INPUT}|head -1 | cut -d "^" -f 3)
echo "The model is ${MODEL}"
```

Shell Script Exercises

4) File: num_of_engines.sh

```
#!/usr/bin/bash
FILE_INPUT=$1
NUM_ENGINES=$2

cut -d "^" -f 7 ${FILE_INPUT}|  grep "${NUM_ENGINES}"| uniq -c | tr -s " " | cut -d " " -f 1
```

- Select the proper column
- Filter rows where the number of engines is the requested
- Count duplicates, and prepare properly the ouput (remove unnecesary blanks and select the last column from the ouptut, maybe there are leading blanks

CSVkit – Exercises 7

1. Use csvstat to find out how many different manufactures are in the file
2. Extract the column manufacturer and using pipes, use sort, uniq and wc find out how many manufacturers are in the file. Why does this number differ to the number reported in csvstat?
3. What are the top 5 manufacturers?
4. Using csvgrep, get only the records with manufacturer equal to *Airbus* and save them to a file with pipe (|) delimiter.

CSVkit – Exercises 7

1) csvstat -d "^" -c manufacturer optd_aircraft.csv

2) csvcut -d '^' -c manufacturer optd_aircraft.csv | tail -n+2 | sort | uniq | wc –l

3) tail -n+2 optd_aircraft.csv | cut -d '^' -f 2 | sort | uniq -c | sort -nr | head –n 5

   or

   csvcut -d '^' -c manufacturer optd_aircraft.csv |csvsort | tail -n+2 | uniq -c |sort -nr | head –n 5

4) csvgrep -d '^' -c manufacturer -m Airbus optd_aircraft.csv | tr "," "|" > airbus.csv

   or

   csvgrep -d '^' -c manufacturer -m Airbus optd_aircraft.csv | csvformat -D '|' > airbus.csv

SQL - Exercises 3

1. Use csvql to query the file: optd_aircraft:
   1. What aircraft model has the most engines?
   2. What is the number of engines more frequent?

```
sqlquery="SELECT * FROM optd_airlines ORDER BY iata_code LIMIT 10"
csvsql --query "$sqlquery" –d “^” opdt_aircraft.csv
```

SQL - Quick exercises 3

select model, nb_engines from optd_aircraft where nb_engines is NOT NULL order by nb_engines DESC limit 1;

```
[dsc@vm:~/Data/opentraveldata] [base] % sql_query="SELECT Model, nb_engines from optd_aircraft WHERE nb_engines IS NOT NULL ORDER BY nb_engines DESC LIMIT 1"
[dsc@vm:~/Data/opentraveldata] [base] % csvsql --query "$sql_query" -d "^" optd_aircraft.csv
model,nb_engines
An-225,6
```

select nb_engines, count(*) from optd_aircraft where nb_engines is NOT NULL group by nb_engines order by count(*) DESC limit 1;

```
[dsc@vm:~/Data/opentraveldata] [base] % sql_query="SELECT nb_engines, COUNT(1) AS n_planes FROM optd_aircraft WHERE nb_engines IS NOT NULL GROUP BY nb_engines
 ORDER BY n_planes DESC LIMIT 1"
[dsc@vm:~/Data/opentraveldata] [base] % csvsql --query "$sql_query" -d "^" optd_aircraft.csv              ~/Data/opentraveldata $COLOR_CONNECTIONS 22:23:42
nb_engines,n_planes
2,241
```