Final Task

CSV CLEANUP

Datetime to date and time

```
#change the space character in the first column to ';', so we will have a column for the date and a column for the time
awk 'BEGIN{FS=OFS=";"} {gsub(" ", ";", $1)} 1' dilans_data.csv > edited_dilans_data.csv
```

text processing - Awk - replace one character only in a certain column - Unix & Linux Stack Exchange

Seperate CSVs by event_type

```
#create new csv file where event_type is read
grep 'read' edited_dilans_data.csv > reader.csv
#from this csv create two file..

  #..one for the first time readers
  grep -E 'Reddit|SEO|AdWords' reader.csv > firsttime_reader.csv  # E for extended regex

  #and one for the returning readers
  grep -v -E 'Reddit|SEO|AdWords' reader.csv > returner.csv #-v for invert selection, -E for extended regex

#create new csv for subscribers
grep 'subscribe' edited_dilans_data.csv > subscriber.csv

#create new csv for purchases
grep 'buy' edited_dilans_data.csv > purchases.csv
```

CREATE TABLES

```
CREATE TABLE first_read (
my_date date,
my_time time,
event_type TEXT,
country TEXT,
user_id TEXT,
source TEXT,
topic TEXT
);

COPY first_read FROM '/home/arthur/final_task/firsttime_reader.csv' DELIMITER ';';
```

```
CREATE TABLE returnings (
my_date date,
my_time time,
event_type TEXT,
country TEXT,
user_id TEXT,
topic TEXT
);

COPY returnings FROM '/home/arthur/final_task/returner.csv' DELIMITER ';';
```

```
CREATE TABLE subscriptions (
my_date date,
my_time time,
event_type TEXT,
user_id TEXT
);

COPY subscriptions FROM '/home/arthur/final_task/subscriber.csv' DELIMITER ';';
```

```
CREATE TABLE purchases (
my_date date,
my_time time,
event_type TEXT,
user_id TEXT,
price INT
);

COPY purchases FROM '/home/arthur/final_task/purchases.csv' DELIMITER ';';
```

PRESENTATION

Slide 3

User funnel by country

```
SELECT first_read.country,
       number_first_readers,
       number_returners,
       number_subscribers,
       number_customers
FROM
  (SELECT country,
          count(distinct(user_id)) AS number_first_readers
   FROM first_read
   GROUP BY country) AS first_read
JOIN
  (SELECT country,
          count(distinct(user_id)) AS number_returners
   FROM returnings
   GROUP BY country) AS returners ON first_read.country = returners.country
JOIN
  (SELECT country,
          count(distinct(subscriptions.user_id))AS number_subscribers
   FROM subscriptions
   JOIN first_read ON subscriptions.user_id = first_read.user_id
   GROUP BY country) AS subscribers ON first_read.country = subscribers.country
JOIN
  (SELECT country,
          count(distinct(purchases.user_id)) AS number_customers
   FROM purchases
   JOIN first_read ON purchases.user_id = first_read.user_id
   GROUP BY country) AS customers ON first_read.country = customers.country;
```

Slide 4

Change in the number of first time readers by country

```
SELECT my_date,
       country,
       Count( user_id )
FROM   first_read
GROUP  BY my_date,
          country;
```

Growth rate of first time readers in the last 3 month by country

```
SELECT januar.country,
       users_in_januar,
       users_in_march,
       (users_in_march::float / users_in_januar)-1 AS growth_rate
FROM
  (SELECT country,
          Count((user_id)) AS users_in_januar
   FROM first_read
   WHERE my_date > '2017-12-31'
     AND my_date < '2018-02-01'
   GROUP BY country) AS januar
JOIN
```

```
  (SELECT country,
          Count((user_id)) AS users_in_march
   FROM first_read
   WHERE my_date > '2018-02-28'
     AND my_date < '2018-04-01'
   GROUP BY country) AS march ON januar.country = march.country
ORDER BY growth_rate DESC;
```

Slide 5

Converstion rate by country

```
SELECT readers.country,
       readers,
       customers,
       (( customers / readers :: FLOAT )) AS conversion_rate
FROM   (SELECT country,
               Count(DISTINCT( user_id )) AS readers
        FROM   first_read
        GROUP  BY country) AS readers
       join (SELECT country,
                    Count(DISTINCT( purchases.user_id )) AS customers
             FROM   purchases
                    join first_read
                      ON purchases.user_id = first_read.user_id
             GROUP  BY country) AS customers
         ON readers.country = customers.country
ORDER  BY conversion_rate DESC;
```

number of users and customers by country

```
SELECT country, Count(*)
FROM   first_read
GROUP  BY country
ORDER  BY Count(*) DESC;
```

```
SELECT country,
       Count(distinct(purchases.user_id)) AS customers
FROM   purchases
       JOIN first_read
         ON purchases.user_id = first_read.user_id
GROUP  BY country
ORDER  BY customers DESC;
```

Joined them together in Google Data Studio to show them in one graph.

Slide 6

Number of readers and revenue

```
SELECT country, Count(*) as number_readers
FROM   first_read
GROUP  BY country
ORDER  BY Count(*) DESC;
```

```
SELECT country,
       Count(country) AS customers,
       Sum(price)     AS revenue
FROM   purchases
       JOIN first_read
         ON purchases.user_id = first_read.user_id
GROUP  BY country
ORDER  BY Sum(price) DESC;
```

Joined them together in Google Data Studio to show them in one graph.

Slide 7

Same as slide 4, +added counted metric in Google Data Studio: ARPU=`revenue`/`number_readers`

Slide 8

Change in the number of purchases

```
SELECT purchases.user_id, purchases.my_date, country
FROM   purchases
         JOIN first_read
         ON purchases.user_id = first_read.user_id;
```

In Google Data Studio the user_id column is counted

Growth rate of purchases

```
SELECT januar.country,
       ( purchases_march :: FLOAT / purchases_january ) - 1 AS growth_rate
FROM   (SELECT country,
               Count(( purchases.user_id )) AS purchases_january
        FROM   purchases
               join first_read
                 ON purchases.user_id = first_read.user_id
        WHERE  purchases.my_date > '2017-12-31'
               AND purchases.my_date < '2018-02-01'
        GROUP  BY country) AS januar
       join (SELECT country,
                    Count(( purchases.user_id )) AS purchases_march
             FROM   purchases
                    join first_read
                      ON purchases.user_id = first_read.user_id
             WHERE  purchases.my_date > '2018-02-28'
                    AND purchases.my_date < '2018-04-01'
             GROUP  BY country) AS march
         ON januar.country = march.country
ORDER  BY growth_rate DESC
```

Slide 11

Net revenue per month

```
#IMPORT CSVS
fr = pd.read_csv('/home/arthur/final_task/firsttime_reader.csv', delimiter = ';', names = ['my_date', 'my_time', 'event_type', 'country', '
purchases = pd.read_csv('/home/arthur/final_task/purchases.csv', delimiter = ';', names = ['my_date', 'my_time', 'event_type', 'user_id','p

#JOIN THE TWO CSVs TOGETHER
purchases_merged = purchases.merge(fr, how = 'left', left_on = 'user_id', right_on = 'user_id')

#FORMAT 'MY_DATE' COLUMN TO DATETIME TYPE SO WE CAN GROUP BY IT LATER
purchases_merged['my_date'] = pd.to_datetime(purchases['my_date'])

#SET 'MY_DATE' COLUMN AS INDEX
purchases_merged = purchases_merged.set_index('my_date')

#ONLY 'PRICE' COLUMN NEEDED, GROUP BY DATE (MONTH) AND ADD THEM TOGETHER
revenue_per_month_by_source = purchases_merged[['price','source']].groupby([pd.Grouper(freq='M'), 'source']).sum()

# RESET INDEX
revenue_per_month_by_source = revenue_per_month_by_source.reset_index(level=0)

revenue_per_month_by_source = revenue_per_month_by_source.reset_index(level=0)

def get_net_revenue(revenue_per_month_by_source):
    if revenue_per_month_by_source['source'] == 'AdWords':
        net_rev =  revenue_per_month_by_source['price'] - (500*3)
    else:
        net_rev =  revenue_per_month_by_source['price'] - (250*3)
    return net_rev

#CREATE NEW COLUMN AND USE DEFINED FUNCTION TO FILL IT UP
revenue_per_month_by_source['net_rev'] = revenue_per_month_by_source.apply(get_net_revenue, axis=1)
```

```
#EXPORT IT TO CSV FILE
revenue_per_month_by_source.to_csv(r'/home/arthur/final_task/net_revenue_per_month_by_source.csv', index=False)
```

Slide 12

Cost per reader

```
#IMPORT CSV
fr = pd.read_csv('/home/arthur/final_task/firsttime_reader.csv', delimiter = ';', names = ['my_date', 'my_time', 'event_type', 'country', '
                                                                 'source', 'topic', ])
#COUNT THE readers BY SOURCE
fr_usercount_by_source = fr.groupby('source').count().sort_values(by=['user_id'], ascending = 0)

#CREATE A FUNCTION FOR LATER USE. THIS FUNCTION GIVES BACK HOW MUCH MONEY WERE SPENT ON EACH USER CONSIDERING THEIR SOURCE
def get_cost_per_reader(fr_usercount_by_source):
    if fr_usercount_by_source['source'] == 'AdWords':
        cost_per_reader = (500*3) / fr_usercount_by_source['user_id'] # ad cost is multiplied by 3 because we look at 3 months
    else:
        cost_per_reader = (250*3) / fr_usercount_by_source['user_id'] # ad cost is multiplied by 3 because we look at 3 months
    return cost_per_reader

#RESET INDEX
fr_usercount_by_source = fr_usercount_by_source.reset_index(level=0)

#CREATE A NEW COLUMN AND EVALUATE IT WITH OUR FUNCTION
fr_usercount_by_source['spent_dollar_on_user'] = fr_usercount_by_source.apply(get_cost_per_reader, axis=1)

#KEEP ONLY THOSE COLUMNS WE NEED
fr_usercount_by_source = fr_usercount_by_source[['source', 'user_id', 'spent_dollar_on_user']]

#EXPORT THE RESULT TO CSV
fr_usercount_by_source.to_csv(r'/home/arthur/final_task/cost_per_reader_by_source.csv', index=False)
```

In Google Data Studio, I multiplied the end value by 1000 to get a more understandable number.

Slide 13

Cost per customer

```
#IMPORT CSVS
fr = pd.read_csv('/home/arthur/final_task/firsttime_reader.csv', delimiter = ';', names = ['my_date', 'my_time', 'event_type', 'country', '
purchases = pd.read_csv('/home/arthur/final_task/purchases.csv', delimiter = ';', names = ['my_date', 'my_time', 'event_type', 'user_id','p

#JOIN THE TWO CSVs TOGETHER
purchases_merged = purchases.merge(fr, how = 'left', left_on = 'user_id', right_on = 'user_id')

#get the unique user_ids
purchases_merged = purchases_merged.drop_duplicates(subset = ["user_id"])

#GROUP BY 'SOURCE' AND COUNT ROWS
purchases_customercount_by_source = purchases_merged.groupby('source').count().sort_values(by=['user_id'], ascending = 0)

#RESET INDEX
purchases_customercount_by_source = purchases_customercount_by_source.reset_index(level=0)

#DEFINE FUNCTION WHICH COUNTS HOW MUCH MONEY WERE SPENT ON CUSTOMER BY SOURCE
def get_cost_per_customer(purchases_customercount_by_source):
    if purchases_customercount_by_source['source'] == 'AdWords':
        cost_per_customer= (500*3) / purchases_customercount_by_source['user_id']
    else:
        cost_per_customer= (250*3) / purchases_customercount_by_source['user_id']
    return cost_per_customer

# CREATE NEW COLUMN AND THEN FILL IT UP USING THE DEFINED FUNCTION
purchases_customercount_by_source['spent_dollar_on_customer'] = purchases_customercount_by_source.apply(get_cost_per_customer, axis=1)

# KEEP ONLY THE NEEDED COLUMNS
purchases_customercount_by_source = purchases_customercount_by_source[['source', 'user_id', 'spent_dollar_on_customer']]

# EXPORT IT TO CSV
purchases_customercount_by_source.to_csv(r'/home/arthur/final_task/cost_per_customer_by_source.csv', index=False)
```

Slide 14

Return of Investment

```
#IMPORT CSVS
fr = pd.read_csv('/home/arthur/final_task/firsttime_reader.csv', delimiter = ';', names = ['my_date', 'my_time', 'event_type', 'country', '
purchases = pd.read_csv('/home/arthur/final_task/purchases.csv', delimiter = ';', names = ['my_date', 'my_time', 'event_type', 'user_id','p

#JOIN THE TWO CSVs TOGETHER
purchases_merged = purchases.merge(fr, how = 'left', left_on = 'user_id', right_on = 'user_id')

#GROUP BY SOURCE AND COUNT SUM OF COLUMNS
revenue_by_source = purchases_merged.groupby('source').sum()

# RESET INDEX
revenue_by_source = revenue_by_source.reset_index(level=0)

#DEFINE FUNCTION WHICH COUNTS THE ROI BY SOURCE
def get_roi(revenue_by_source):
    if revenue_by_source['source'] == 'AdWords':
        roi =  (revenue_by_source['price']-500*3)/(500*3)
    else:
        roi =  (revenue_by_source['price']-250*3)/(250*3)
    return roi

#CREATE NEW COLUMN AND USE DEFINED FUNCTION TO FILL IT UP
revenue_by_source['roi'] = revenue_by_source.apply(get_roi, axis=1)

#KEEP ONLY THE COLUMNS THAT ARE NEEDED
roi_by_source = revenue_by_source[['source', 'price', 'roi']]

#EXPORT TO CSV
roi_by_source.to_csv(r'/home/arthur/final_task/roi.csv', index=False)
```

Slide 15

Topics by country

```
select  topic, country, count(user_id) from first_read
group by  country, topic
order by country, count(user_id) desc;
```

Slide 16

Topics by source

```
# IMPORT CSV
fr = pd.read_csv('/home/arthur/final_task/firsttime_reader.csv', delimiter = ';', names = ['my_date', 'my_time', 'event_type', 'country', '
                                                                 'source', 'topic', ])

#SELECT NEEDED COLUMNS AND GROUP BY COUNTRY, SOURCE, TOPIC AND COUNT ROWS
users_by_country_by_source_by_topic = fr[['country','source','user_id','topic']].groupby(['country', 'source', 'topic']).count()

#EXPORT TO CSV
users_by_country_by_source_by_topic.to_csv(r'/home/arthur/final_task/users_by_country_by_source_by_topic.csv', index=True)
```