Data Analysis with Python

Credit Models

Presents :

- José Armando Melchor Soto

Libraries

In [44]:
```
import pandas as pd
```

Data Import

In [45]:
```
visits = pd.read_csv('visits-1.csv',
                     parse_dates=[1])

cart = pd.read_csv('cart-1.csv',
                   parse_dates=[1])

checkout = pd.read_csv('checkout-1.csv',
                       parse_dates=[1])

purchase = pd.read_csv('purchase-1.csv',
                       parse_dates=[1])
```

Exercises

Step 1: Inspect the DataFrames using `print` and `head`

In [46]:
```
display(visits.head())
display(cart.head())
display(checkout.head())
display(purchase.head())
```

	user_id	visit_time
0	943647ef-3682-4750-a2e1-918ba6f16188	2017-04-07 15:14:00
1	0c3a3dd0-fb64-4eac-bf84-ba069ce409f2	2017-01-26 14:24:00
2	6e0b2d60-4027-4d9a-babd-0e7d40859fb1	2017-08-20 08:23:00
3	6879527e-c5a6-4d14-b2da-50b85212b0ab	2017-11-04 18:15:00
4	a84327ff-5daa-4ba1-b789-d5b4caf81e96	2017-02-27 11:25:00

	user_id	cart_time
0	2be90e7c-9cca-44e0-bcc5-124b945ff168	2017-11-07 20:45:00
1	4397f73f-1da3-4ab3-91af-762792e25973	2017-05-27 01:35:00
2	a9db3d4b-0a0a-4398-a55a-ebb2c7adf663	2017-03-04 10:38:00
3	b594862a-36c5-47d5-b818-6e9512b939b3	2017-09-27 08:22:00
4	a68a16e2-94f0-4ce8-8ce3-784af0bbb974	2017-07-26 15:48:00

	user_id	checkout_time
0	d33bdc47-4afa-45bc-b4e4-dbe948e34c0d	2017-06-25 09:29:00
1	4ac186f0-9954-4fea-8a27-c081e428e34e	2017-04-07 20:11:00
2	3c9c78a7-124a-4b77-8d2e-e1926e011e7d	2017-07-13 11:38:00
3	89fe330a-8966-4756-8f7c-3bdbcd47279a	2017-04-20 16:15:00
4	3ccdaf69-2d30-40de-b083-51372881aedd	2017-01-08 20:52:00

	user_id	purchase_time
0	4b44ace4-2721-47a0-b24b-15fbfa2abf85	2017-05-11 04:25:00
1	02e684ae-a448-408f-a9ff-dcb4a5c99aac	2017-09-05 08:45:00
2	4b4bc391-749e-4b90-ab8f-4f6e3c84d6dc	2017-11-20 20:49:00
3	a5dbb25f-3c36-4103-9030-9f7c6241cd8d	2017-01-22 15:18:00
4	46a3186d-7f5a-4ab9-87af-84d05bfd4867	2017-06-11 11:32:00

Step 2: Left merging visits and cart

In [47]:
```
visits_car = visits.merge(cart, how='left', on='user_id')
visits_car
#Left merge -> visits.me.....
```

Out[47]:

	user_id	visit_time	cart_time
0	943647ef-3682-4750-a2e1-918ba6f16188	2017-04-07 15:14:00	NaT
1	0c3a3dd0-fb64-4eac-bf84-ba069ce409f2	2017-01-26 14:24:00	2017-01-26 14:44:00
2	6e0b2d60-4027-4d9a-babd-0e7d40859fb1	2017-08-20 08:23:00	2017-08-20 08:31:00
3	6879527e-c5a6-4d14-b2da-50b85212b0ab	2017-11-04 18:15:00	NaT
4	a84327ff-5daa-4ba1-b789-d5b4caf81e96	2017-02-27 11:25:00	NaT
...	...	...	...
1995	33913ac2-03da-45ae-8fc3-fea39df827c6	2017-03-25 03:29:00	NaT
1996	4f850132-b99d-4623-80e6-6e61d003577e	2017-01-08 09:57:00	NaT
1997	f0830b9b-1f5c-4e74-b63d-3f847cc6ce70	2017-09-07 12:56:00	NaT
1998	b01bffa7-63ba-4cd3-9d93-eb1477c23831	2017-07-20 04:37:00	NaT
1999	0336ca81-8d68-443f-9248-ac0b8ad147d5	2017-11-15 10:11:00	NaT

2000 rows × 3 columns

Step 3: How long is `visits_cart`?

In [48]:
```
visits_car.shape
```

Out[48]:
```
(2000, 3)
```

Step 4: How many timestamps are null for `cart_time`?

In [49]:
```
null_cart_time = len(visits_car[visits_car['cart_time'].isnull()])
null_cart_time
# Empty cart time
```

Out[49]:
```
1652
```

Step 5: What percentage only visited?

In [50]:
```
visited_not_cart = (null_cart_time / visits_car.shape[0]) * 100
print(f'{visited_not_cart} %')
# use the null carts and the total
```

```
82.6 %
```

Step 6: What percentage placed a t-shirt in their cart but did not checkout?

In [51]:
```
cart_checkout = cart.merge(checkout, how = 'left', on = 'user_id')

# left merge cart and checkout
no_checkout = cart_checkout[cart_checkout['checkout_time'].isnull()]

percentage = len(no_checkout) / len(cart_checkout) * 100

print(f" {percentage:.2f}% of shoppers placed a t-shirt in their cart but did no
```

```
35.06% of shoppers placed a t-shirt in their cart but did not checkout.
```

Step 7: Merge it all together

In [54]:
```
# left merge and left merge...
all_data = (
    visits
    .merge(cart, how='left', on='user_id')
    .merge(checkout, how='left', on='user_id')
    .merge(purchase, how='left', on='user_id')
)

all_data.head()
```

Out[54]:

	user_id	visit_time	cart_time	checkout_time	purchase_time
0	943647ef-3682-4750-a2e1-918ba6f16188	2017-04-07 15:14:00	NaT	NaT	NaT
1	0c3a3dd0-fb64-4eac-bf84-ba069ce409f2	2017-01-26 14:24:00	2017-01-26 14:44:00	2017-01-26 14:54:00	2017-01-26 15:08:00
2	6e0b2d60-4027-4d9a-babd-0e7d40859fb1	2017-08-20 08:23:00	2017-08-20 08:31:00	NaT	NaT
3	6879527e-c5a6-4d14-b2da-50b85212b0ab	2017-11-04 18:15:00	NaT	NaT	NaT
4	a84327ff-5daa-4ba1-b789-d5b4caf81e96	2017-02-27 11:25:00	NaT	NaT	NaT

Step 8: % of users who got to checkout but did not purchase

In [72]:
```
percentage_checkout = (
    all_data[all_data['checkout_time'].notna()]['purchase_time']
    .isna()
    .mean() * 100
)

print(f'{percentage_checkout:.2f} %')
```

```
24.55 %
```

Step 9: check each part of the funnel, let's print all 3 of them again

In [85]:
```
percentage_1 = (
    all_data['cart_time']
    .isna()
    .mean() * 100
)
```

```
percentage_2 = (
    all_data[all_data['cart_time'].notna()]['checkout_time']
    .isna()
    .mean() * 100
)

percentage_3 = (
    all_data[all_data['checkout_time'].notna()]['purchase_time']
    .isna()
    .mean() * 100
)
```

In [86]:
```
print(f"{percentage_1:.2f} % of users who visited the page did not add a t-shirt
print(f"{percentage_2:.2f} % of users who added a t-shirt to their cart did not
print(f"{percentage_3:.2f} % of users who made it to checkout  did not purchase
```

```
78.37 % of users who visited the page did not add a t-shirt to their cart
26.75 % of users who added a t-shirt to their cart did not checkout
24.55 % of users who made it to checkout  did not purchase a shirt
```

The weakest part of the funnel is clearly getting a person who visited the site to add a tshirt to their cart. Once they've added a t-shirt to their cart it is fairly likely they end up purchasing it. A suggestion could be to make the add-to-cart button more prominent on the front page.

Step 10: adding new column

In [90]:
```
all_data['time_to_purchase'] = all_data.purchase_time - all_data.visit_time
```

Step 11: examine the results

In [89]:
```
all_data.head()
```

Out[89]:

	user_id	visit_time	cart_time	checkout_time	purchase_time	time_to_purchase
0	943647ef-3682-4750-a2e1-918ba6f16188	2017-04-07 15:14:00	NaT	NaT	NaT	NaT
1	0c3a3dd0-fb64-4eac-bf84-ba069ce409f2	2017-01-26 14:24:00	2017-01-26 14:44:00	2017-01-26 14:54:00	2017-01-26 15:08:00	0 days 00:44:00
2	6e0b2d60-4027-4d9a-babd-0e7d40859fb1	2017-08-20 08:23:00	2017-08-20 08:31:00	NaT	NaT	NaT
3	6879527e-c5a6-4d14-b2da-50b85212b0ab	2017-11-04 18:15:00	NaT	NaT	NaT	NaT
4	a84327ff-5daa-4ba1-b789-d5b4caf81e96	2017-02-27 11:25:00	NaT	NaT	NaT	NaT

Step 12: what is the average time to purchase?

In [91]:
```
all_data['time_to_purchase'].mean()
```

Out[91]:
```
Timedelta('0 days 00:43:12.380952380')
```