

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4 import plotly.express as px
5 import plotly.graph_objects as go
6 from plotly.subplots import make_subplots
```

✓ Data loading and exploring

```
1 # data converted to csv as while loading in excel taking too much
2 df_es = pd.read_csv('/content/ES.csv')
3 df_fr = pd.read_csv('/content/FR.csv')
4 df_uk = pd.read_csv('/content/UK.csv')
5 df_de = pd.read_csv('/content/DE.csv')
6 df_us = pd.read_csv('/content/US.csv')
7
8 print(df_es.head(2))
9 print(df_fr.head(2))
10 print(df_uk.head(2))
11 print(df_de.head(2))
12 print(df_us.head(2))
```



0	0.0	0.0	0.0
1	0.0	0.0	0.0

	selling fees	fba fees	other transaction fees	other	total
0	1.68	0.0	0.0	0.0	-12.31
1	0.00	0.0	0.0	6.46	6.46

[2 rows x 28 columns]

	date/time	settlement id	type	order id	\
0	Jan 1, 2021 1:24:15 AM PST	13704186161	Adjustment	114-3447667-8401830	
1	Jan 1, 2021 2:56:03 AM PST	13704186161	Adjustment	114-6813146-6150605	

	sku	description	\
0	PAN-EU-App-PhoRep-1274	FBA Inventory Reimbursement - Customer Return	
1	GR-J04V-42V3	FBA Inventory Reimbursement - Customer Return	

	quantity	marketplace	Country	fulfilment	...	promotional rebates	tax	\
0	1.0	NaN	US	Standard Orders	...		0	
1	1.0	NaN	US	Standard Orders	...		0	

	marketplace withheld tax	selling fees	fba fees	other transaction fees	other	\
0	0	0.0	0		0.0 0.0	
1	0	0.0	0		0.0 0.0	

	total	other transaction fees.1	other.1	total.1
0	0.0	0.0	11.09	11.09
1	0.0	0.0	5.14	5.14

[2 rows x 31 columns]

```
1 print(df_es.shape)
2 print(df_fr.shape)
3 print(df_uk.shape)
4 print(df_de.shape)
5 print(df_us.shape)
```

```
➡ (266201, 28)
(531454, 28)
(433052, 28)
(781300, 28)
(706797, 31)
```

```
1 print(df_es.info())
2 print(df_fr.info())
3 print(df_uk.info())
4 print(df_de.info())
5 print(df_us.info())
```

```
➡ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 266201 entries, 0 to 266200
Data columns (total 28 columns):
#   Column                Non-Null Count  Dtype
---  -
0   date/time              266201 non-null object
```

```

1  settlement id      266201 non-null  int64
2  type               266201 non-null  object
3  order id          261820 non-null  object
4  sku                263626 non-null  object
5  description        266186 non-null  object
6  quantity           263384 non-null  float64
7  marketplace        258443 non-null  object
8  Country            266201 non-null  object
9  fulfilment         256012 non-null  object
10 order city         256010 non-null  object
11 order state        250154 non-null  object
12 order postal       256008 non-null  object
13 tax collection model 54 non-null    object
14 product sales      266201 non-null  float64
15 product sales tax  266201 non-null  float64
16 postage credits    266201 non-null  float64
17 shipping credits tax 266201 non-null  float64
18 gift wrap credits  266201 non-null  float64
19 giftwrap credits tax 266201 non-null  float64
20 promotional rebates 266201 non-null  float64
21 promotional rebates tax 266201 non-null  float64
22 marketplace withheld tax 266201 non-null  float64
23 selling fees       266201 non-null  float64
24 fba fees           266201 non-null  float64
25 other transaction fees 266201 non-null  float64
26 other              266201 non-null  float64
27 total              266201 non-null  float64

```

dtypes: float64(15), int64(1), object(12)

memory usage: 56.9+ MB

None

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 531454 entries, 0 to 531453

Data columns (total 28 columns):

#	Column	Non-Null Count	Dtype
0	date/time	531454 non-null	object
1	settlement id	531454 non-null	int64
2	type	531454 non-null	object
3	order id	526558 non-null	object
4	sku	527757 non-null	object
5	description	531441 non-null	object
6	quantity	527318 non-null	float64
7	marketplace	518301 non-null	object
8	Country	531454 non-null	object
9	fulfilment	514900 non-null	object
10	order city	514897 non-null	object
11	order state	70558 non-null	object
12	order postal	514884 non-null	object
13	tax collection model	40 non-null	object
14	product sales	531454 non-null	float64
15	product sales tax	531454 non-null	float64
16	postage credits	531454 non-null	float64

```

1 print(df_es.describe())
2 print(df_fr.describe())
3 print(df_uk.describe())
4 print(df_de.describe())
5 print(df_us.describe())

```



	settlement id	quantity	product sales	product sales tax \
count	2.662010e+05	263384.000000	266201.000000	266201.000000
mean	1.791711e+10	4.931841	7.820112	1.482195
std	1.834035e+09	3.164997	9.600997	1.898482
min	1.465180e+10	0.000000	-141.240000	-29.640000
25%	1.642754e+10	2.000000	6.600000	1.390000
50%	1.793293e+10	5.000000	7.430000	1.560000
75%	1.929530e+10	8.000000	10.730000	2.080000
max	2.134899e+10	10.000000	236.900000	46.800000

	postage credits	shipping credits tax	gift wrap credits \
count	266201.000000	266201.000000	266201.000000
mean	0.197646	0.036783	0.000359
std	0.864099	0.167286	0.035671
min	-13.580000	-2.520000	-3.300000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000
max	58.800000	3.350000	3.990000

	giftwrap credits tax	promotional rebates	promotional rebates tax \
count	266201.000000	266201.000000	266201.000000
mean	0.000072	-0.037739	-0.007024
std	0.007299	0.268605	0.053622
min	-0.690000	-6.120000	-1.280000
25%	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000
max	0.690000	3.990000	0.720000

	marketplace withheld tax	selling fees	fba fees \
count	266201.000000	266201.000000	266201.000000
mean	-0.00042	-1.460285	-2.597334
std	0.03500	1.652376	1.643336
min	-5.46000	-57.600000	-100.000000
25%	0.00000	-1.950000	-3.020000
50%	0.00000	-1.390000	-2.510000
75%	0.00000	-1.230000	-2.260000
max	1.87000	16.600000	11.040000

	other transaction fees	other	total
count	266201.000000	266201.000000	266201.000000
mean	-0.704205	0.038845	4.769006
std	18.800518	4.135264	21.367608
min	-524.240000	-614.740000	-614.740000
25%	0.000000	0.000000	4.300000
50%	0.000000	0.000000	5.180000
75%	0.000000	0.000000	7.800000
max	0.000000	180.000000	212.190000

	settlement id	quantity	product sales	product sales tax \
count	5.314540e+05	527318.000000	531454.000000	531454.000000
mean	1.804464e+10	9.470039	8.458621	1.610610
std	1.854689e+09	6.064791	8.386757	1.624145
min	1.464804e+10	0.000000	-129.990000	-26.000000
25%	1.651081e+10	4.000000	6.660000	1.330000
50%	1.820392e+10	9.000000	8.320000	1.670000
75%	1.955005e+10	15.000000	10.820000	2.170000
max	2.134465e+10	20.000000	403.650000	55.440000

```
1 print(df_es.isnull().sum())  
2 print(df_fr.isnull().sum())  
3 print(df_uk.isnull().sum())  
4 print(df_de.isnull().sum())  
5 print(df_us.isnull().sum())
```



```

other          0
total          0
other transaction fees.1  0
other.1        0
total.1        0
dtype: int64

```

```

1 print(df_es.duplicated().sum())
2 print(df_fr.duplicated().sum())
3 print(df_uk.duplicated().sum())
4 print(df_de.duplicated().sum())
5 print(df_us.duplicated().sum())

```

```

719
490
2976
4353
5690

```

```
1 print()
```




✓ Data cleaning and preprocessing

```

1 def clean_and_preprocess(df, country_code, exchange_rates):
2     if country_code in ['ES', 'FR', 'UK']:
3         df['date/time'] = pd.to_datetime(df['date/time'], format=
4     elif country_code == 'DE':
5         df['date/time'] = pd.to_datetime(df['date/time'], format=
6     elif country_code == 'US':
7         df['date/time'] = pd.to_datetime(df['date/time'], format=
8
9     df = df.dropna(subset=['total'])
10    df = df.drop_duplicates()
11
12    df['country'] = country_code
13    df['total'] = pd.to_numeric(df['total'], errors='coerce')
14
15    if country_code in exchange_rates:
16        df['total_eur'] = df['total'] * exchange_rates[country_co
17    else:
18        df['total_eur'] = df['total']
19
20    return df
21
22

```

```
23 exchange_rates = {
24     'UK': 1.18,
25     'US': 0.93,
26     'DE': 1.00,
27     'FR': 1.00,
28     'ES': 1.00
29 }
30
31 df_es_cleaned = clean_and_preprocess(df_es.copy(), 'ES', exchange
32 df_fr_cleaned = clean_and_preprocess(df_fr.copy(), 'FR', exchange
33 df_uk_cleaned = clean_and_preprocess(df_uk.copy(), 'UK', exchange
34 df_de_cleaned = clean_and_preprocess(df_de.copy(), 'DE', exchange
35 df_us_cleaned = clean_and_preprocess(df_us.copy(), 'US', exchange
36
37 print("Done cleaning and prepping all datasets.")
38
```

 Done cleaning and prepping all datasets.

✓ Combine data

```
1 dataframes_to_combine = [df_de_cleaned, df_es_cleaned, df_fr_clea
2 df_combined = pd.concat(dataframes_to_combine, ignore_index=True)
3
4 display(df_combined.head())
5 print(df_combined.shape)
```



	date/time	settlement id	type	order id	sku	description	quantity	marke
0	2020-12-31 23:54:41	14668686782	Refund	305-3488972-4981168	PAN-EU-App-PhoRep-1151 Child2	MMOBIEL Akku kompatibel mit iPhone 4S Li-Ion B...	9.0	amæ
1	2021-01-01 00:25:44	14668686782	Adjustment	305-3760164-5117930	PAN-EU-MM-Tool-1017	Versand durch Amazon Erstattung für Lagerbesta...	3.0	
2	2021-01-01 00:26:05	14668686782	Adjustment	302-2179035-0298738	PAN-EU-Sam-PhoRep-1215	Versand durch Amazon Erstattung für Lagerbesta...	7.0	
3	2021-01-01 01:02:31	14668686782	Adjustment	NaN	PAN-EU-App-PhoRep-1008	Versand durch Amazon Erstattung für Lagerbesta...	3.0	
4	2021-01-01 02:38:31	14668686782	Adjustment	NaN	PAN-EU-Sam-PhoRep-1311 Child1	Versand durch Amazon Erstattung für Lagerbesta...	4.0	

5 rows × 33 columns

(2697990, 33)

✓ Data analysis and reporting

Subtask:

Perform data analysis to determine sales by country and region, analyze trends over time, and investigate refunds.

Reasoning: Perform the data analysis steps requested in the subtask: calculate total sales by country and type, group by month for time trend analysis, and calculate total refund amount by country.

1

```
2 def summarize_total(df, group_col, value_col='total_eur', filter_
```



```

3     if filter_cond is not None:
4         df = df.query(filter_cond)
5     summary = df.groupby(group_col)[value_col].sum().reset_index()
6     return summary
7
8 df_combined['month'] = df_combined['date/time'].dt.to_period('M')
9
10 sales_by_country = summarize_total(df_combined, 'country')
11 sales_by_type = summarize_total(df_combined, 'type')
12 monthly_sales = summarize_total(df_combined, 'month')
13 refunds_by_country = summarize_total(df_combined, 'country', filt
14
15 print("Total Sales by Country:")
16 print(sales_by_country)
17
18 print("\nTotal Sales by Transaction Type:")
19 print(sales_by_type)
20
21 print("\nMonthly Sales Trends:")
22 print(monthly_sales)
23
24 print("\nTotal Refund Amount by Country:")
25 print(refunds_by_country)

```

```

26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

Monthly Sales Trends:

```

16 2022-04 1.298311e+05
17 2022-05 1.655399e+05
18 2022-06 1.720682e+05
19 2022-07 1.880690e+05
20 2022-08 1.580547e+05
21 2022-09 1.147745e+05
22 2022-10 2.024582e+05
23 2022-11 2.019905e+05
24 2022-12 1.857764e+05
25 2023-01 1.992102e+05
26 2023-02 1.983045e+05
27 2023-03 2.167420e+05
28 2023-04 1.686332e+05
29 2023-05 2.059485e+05
30 2023-06 1.813801e+05
31 2023-07 2.033312e+05
32 2023-08 1.688099e+05
33 2023-09 1.084340e+05
34 2023-10 2.024556e+05
35 2023-11 1.937571e+05
36 2023-12 1.761942e+05
37      NaT 1.414322e+06

```

Total Refund Amount by Country:

	country	total_eur
0	DE	-672532.0900
1	ES	-280885.7200
2	FR	-425004.8100
3	UK	-179472.0292
4	US	20254.2654

✓ Visualization

```

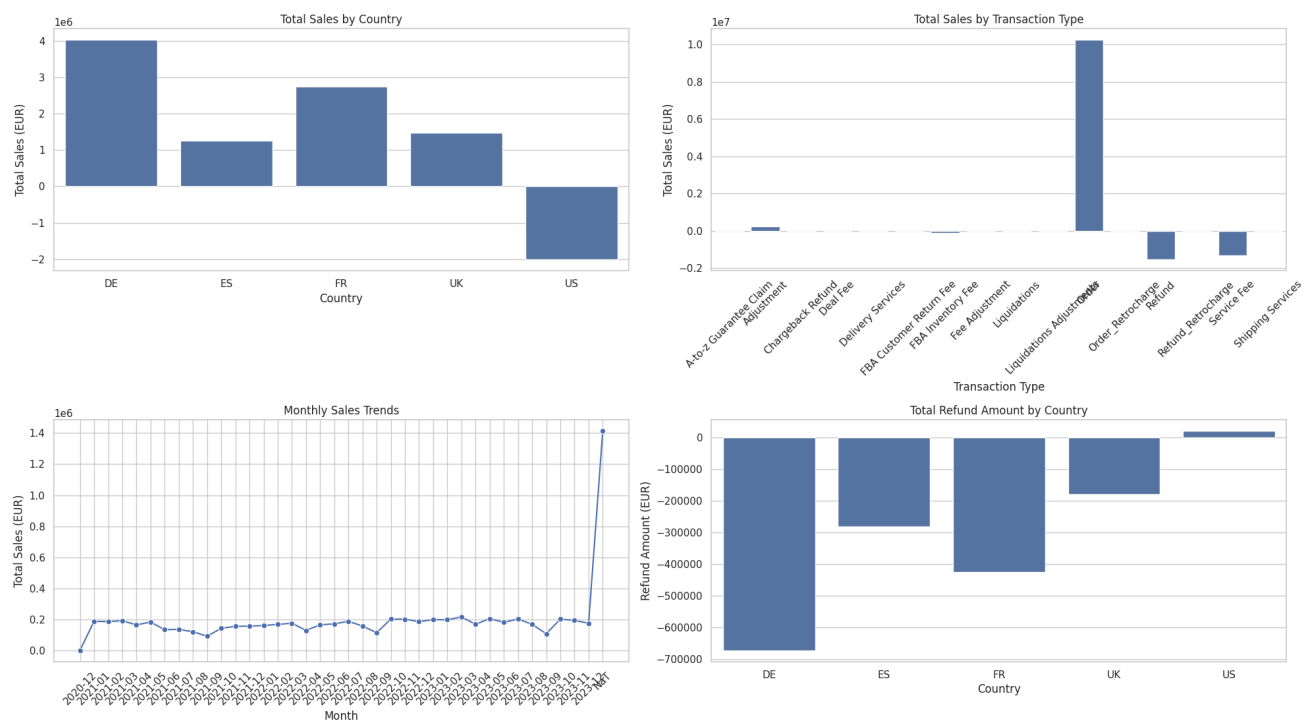
1 sns.set(style='whitegrid')
2
3 fig, axs = plt.subplots(2, 2, figsize=(20, 12))
4 fig.suptitle('Sales Analysis Overview', fontsize=18, fontweight='
5
6 # Chart 1: Total Sales by Country
7 sns.barplot(x='country', y='total_eur', data=sales_by_country, ax
8 axs[0, 0].set_title('Total Sales by Country')
9 axs[0, 0].set_xlabel('Country')
10 axs[0, 0].set_ylabel('Total Sales (EUR)')
11
12 # Chart 2: Total Sales by Transaction Type
13 sns.barplot(x='type', y='total_eur', data=sales_by_type, ax=axs[0
14 axs[0, 1].set_title('Total Sales by Transaction Type')
15 axs[0, 1].set_xlabel('Transaction Type')
16 axs[0, 1].set_ylabel('Total Sales (EUR)')
17 axs[0, 1].tick_params(axis='x', rotation=45)
18
19 # Chart 3: Monthly Sales Trends

```

```
20 sns.lineplot(x='month', y='total_eur', data=monthly_sales, marker
21 axs[1, 0].set_title('Monthly Sales Trends')
22 axs[1, 0].set_xlabel('Month')
23 axs[1, 0].set_ylabel('Total Sales (EUR)')
24 axs[1, 0].tick_params(axis='x', rotation=45)
25
26 # Chart 4: Total Refund Amount by Country
27 sns.barplot(x='country', y='total_eur', data=refunds_by_country,
28 axs[1, 1].set_title('Total Refund Amount by Country')
29 axs[1, 1].set_xlabel('Country')
30 axs[1, 1].set_ylabel('Refund Amount (EUR)')
31
32 plt.tight_layout(rect=[0, 0, 1, 0.96])
33 plt.show()
34
```



Sales Analysis Overview



```

1 fig = make_subplots(
2     rows=2, cols=2,
3     subplot_titles=[
4         "Total Sales by Country",
5         "Total Sales by Transaction Type",
6         "Monthly Sales Trends",
7         "Total Refund Amount by Country"

```

```
8     ]
9 )
10
11 # Chart 1: Total Sales by Country
12 fig.add_trace(
13     go.Bar(
14         x=sales_by_country['country'],
15         y=sales_by_country['total_eur'],
16         name="Sales by Country",
17         marker_color='royalblue'
18     ),
19     row=1, col=1
20 )
21
22 # Chart 2: Total Sales by Transaction Type
23 fig.add_trace(
24     go.Bar(
25         x=sales_by_type['type'],
26         y=sales_by_type['total_eur'],
27         name="Sales by Type",
28         marker_color='seagreen'
29     ),
30     row=1, col=2
31 )
32
33 # Chart 3: Monthly Sales Trends
34 fig.add_trace(
35     go.Scatter(
36         x=monthly_sales['month'],
37         y=monthly_sales['total_eur'],
38         mode='lines+markers',
39         name="Monthly Sales",
40         line=dict(color='orange', width=3)
41     ),
42     row=2, col=1
43 )
44
45 # Chart 4: Total Refund Amount by Country
46 fig.add_trace(
47     go.Bar(
48         x=refunds_by_country['country'],
49         y=refunds_by_country['total_eur'],
50         name="Refunds by Country",
51         marker_color='crimson'
52     ),
```

```
53     row=2, col=2
54 )
55
56 # Update layout
57 fig.update_layout(
58     height=800,
59     width=1200,
60     title_text="Sales Analytics Dashboard"
61     showlegend=False,
62     template='plotly_white',
63     margin=dict(t=80, b=50)
64 )
65
66 fig.show()
```

