sql-capstone-project-upload

July 3, 2024

[3]:
```python
import pymysql
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import warnings
import plotly.graph_objects as go

warnings.filterwarnings('ignore')

# Connecting to MySQL database in MySql Workbench using pymysql
conn = pymysql.connect(
       host='localhost',
       user='root',
       password='12345678',
       db='sql_capstone_project'
)

cur = conn.cursor()
```

[4]:
```python
#Retrieving the given dataset
query1 = 'SELECT * FROM amazon'
cur.execute(query1)
out1 = cur.fetchall()
# Creating DataFrame using Pandas
df1 = pd.DataFrame(out1, columns=['invoice_id', 'branch',␣
 ↪'city','customer_type',
'gender','product_line','unit_price','quantity','VAT','total','date','time','payment_method','
,'gross_margin_percentage','gross_income','rating'])
df1
```

[4]:
```
     invoice_id branch       city customer_type  gender  \
0   750-67-8428      A     Yangon        Member  Female
1   226-31-3081      C  Naypyitaw        Normal  Female
2   631-41-3108      A     Yangon        Normal    Male
3   123-19-1176      A     Yangon        Member    Male
4   373-73-7910      A     Yangon        Normal    Male
```

```
..           …    …         …          …     …
995  233-67-5758      C  Naypyitaw       Normal    Male
996  303-96-2227      B   Mandalay       Normal  Female
997  727-02-1313      A     Yangon       Member    Male
998  347-56-2442      A     Yangon       Normal    Male
999  849-09-3807      A     Yangon       Member  Female

               product_line  unit_price  quantity      VAT      total  \
0          Health and beauty       74.69         7  26.1415   548.9715
1     Electronic accessories       15.28         5   3.8200    80.2200
2         Home and lifestyle       46.33         7  16.2155   340.5255
3          Health and beauty       58.22         8  23.2880   489.0480
4          Sports and travel       86.31         7  30.2085   634.3785
..                       …           …         …        …          …
995        Health and beauty       40.35         1   2.0175    42.3675
996       Home and lifestyle       97.38        10  48.6900  1022.4900
997      Food and beverages        31.84         1   1.5920    33.4320
998       Home and lifestyle       65.82         1   3.2910    69.1110
999     Fashion accessories       88.34         7  30.9190   649.2990

           date      time payment_method    cogs  gross_margin_percentage  \
0    2019-01-05  13:08:00        Ewallet  522.83                 4.761905
1    2019-03-08  10:29:00           Cash   76.40                 4.761905
2    2019-03-03  13:23:00    Credit card  324.31                 4.761905
3    2019-01-27  20:33:00        Ewallet  465.76                 4.761905
4    2019-02-08  10:37:00        Ewallet  604.17                 4.761905
..          …         …              …       …                        …
995  2019-01-29  13:46:00        Ewallet   40.35                 4.761905
996  2019-03-02  17:16:00        Ewallet  973.80                 4.761905
997  2019-02-09  13:22:00           Cash   31.84                 4.761905
998  2019-02-22  15:33:00           Cash   65.82                 4.761905
999  2019-02-18  13:28:00           Cash  618.38                 4.761905

     gross_income  rating
0         26.1415     9.1
1          3.8200     9.6
2         16.2155     7.4
3         23.2880     8.4
4         30.2085     5.3
..             …       …
995        2.0175     6.2
996       48.6900     4.4
997        1.5920     7.7
998        3.2910     4.1
999       30.9190     6.6

[1000 rows x 17 columns]
```

```
[5]: query1 = 'SET SQL_SAFE_UPDATES = 0'
     cur.execute(query1)
     out1 = cur.fetchall()
```

```
[6]: # Adding timeofday column
     query1 = 'ALTER TABLE amazon ADD COLUMN timeofday VARCHAR(20)'
     cur.execute(query1)
     query2 = 'SELECT * FROM amazon'
     cur.execute(query2)
     out1=cur.fetchall()
     df1 = pd.DataFrame(out1, columns=['invoice_id', 'branch',␣
       ↪'city','customer_type',
     'gender','product_line','unit_price','quantity','VAT','total','date','time','payment_method','
     ,'gross_margin_percentage','gross_income','rating','timeofday'])
     df1
```

```
[6]:       invoice_id branch       city customer_type  gender  \
     0    750-67-8428      A     Yangon        Member  Female
     1    226-31-3081      C  Naypyitaw        Normal  Female
     2    631-41-3108      A     Yangon        Normal    Male
     3    123-19-1176      A     Yangon        Member    Male
     4    373-73-7910      A     Yangon        Normal    Male
     ..           …     …        …             …      …
     995  233-67-5758      C  Naypyitaw        Normal    Male
     996  303-96-2227      B   Mandalay        Normal  Female
     997  727-02-1313      A     Yangon        Member    Male
     998  347-56-2442      A     Yangon        Normal    Male
     999  849-09-3807      A     Yangon        Member  Female

                    product_line  unit_price  quantity      VAT      total  \
     0         Health and beauty       74.69         7  26.1415   548.9715
     1    Electronic accessories       15.28         5   3.8200    80.2200
     2        Home and lifestyle       46.33         7  16.2155   340.5255
     3         Health and beauty       58.22         8  23.2880   489.0480
     4         Sports and travel       86.31         7  30.2085   634.3785
     ..                      …         …       …       …         …
     995       Health and beauty       40.35         1   2.0175    42.3675
     996      Home and lifestyle       97.38        10  48.6900  1022.4900
     997      Food and beverages       31.84         1   1.5920    33.4320
     998      Home and lifestyle       65.82         1   3.2910    69.1110
     999     Fashion accessories       88.34         7  30.9190   649.2990

                date      time payment_method    cogs  gross_margin_percentage  \
     0    2019-01-05  13:08:00        Ewallet  522.83                 4.761905
     1    2019-03-08  10:29:00           Cash   76.40                 4.761905
     2    2019-03-03  13:23:00    Credit card  324.31                 4.761905
     3    2019-01-27  20:33:00        Ewallet  465.76                 4.761905
```

```
4    2019-02-08  10:37:00        Ewallet  604.17                 4.761905
..          …         …          …      …                       …
995  2019-01-29  13:46:00        Ewallet   40.35                 4.761905
996  2019-03-02  17:16:00        Ewallet  973.80                 4.761905
997  2019-02-09  13:22:00           Cash   31.84                 4.761905
998  2019-02-22  15:33:00           Cash   65.82                 4.761905
999  2019-02-18  13:28:00           Cash  618.38                 4.761905

     gross_income  rating timeofday
0         26.1415     9.1      None
1          3.8200     9.6      None
2         16.2155     7.4      None
3         23.2880     8.4      None
4         30.2085     5.3      None
..            …       …         …
995        2.0175     6.2      None
996       48.6900     4.4      None
997        1.5920     7.7      None
998        3.2910     4.1      None
999       30.9190     6.6      None

[1000 rows x 18 columns]
```

[7]:
```
query1="""UPDATE amazon
SET timeofday = CASE
    WHEN HOUR(time) >=6 AND HOUR(time) < 12 THEN 'Morning'
    WHEN HOUR(time) >=12 AND HOUR(time) < 18 THEN 'Afternoon'
    WHEN HOUR(time) >=18 AND HOUR(time) < 24 THEN 'Evening'
    ELSE 'Night'
    END"""
cur.execute(query1)
query2 = 'SELECT * FROM amazon'
cur.execute(query2)
out2=cur.fetchall()
df1 = pd.DataFrame(out2,columns=['invoice_id', 'branch', 'city','customer_type',
'gender','product_line','unit_price','quantity','VAT','total','date','time','payment_method','
,'gross_margin_percentage','gross_income','rating','timeofday'])
df1
```

[7]:
```
      invoice_id branch       city customer_type  gender  \
0    750-67-8428      A     Yangon        Member  Female
1    226-31-3081      C  Naypyitaw        Normal  Female
2    631-41-3108      A     Yangon        Normal    Male
3    123-19-1176      A     Yangon        Member    Male
4    373-73-7910      A     Yangon        Normal    Male
..           …     …         …           …     …
995  233-67-5758      C  Naypyitaw        Normal    Male
```

```
996  303-96-2227      B   Mandalay        Normal  Female
997  727-02-1313      A     Yangon        Member    Male
998  347-56-2442      A     Yangon        Normal    Male
999  849-09-3807      A     Yangon        Member  Female

               product_line  unit_price  quantity      VAT      total  \
0         Health and beauty       74.69         7  26.1415   548.9715
1    Electronic accessories       15.28         5   3.8200    80.2200
2        Home and lifestyle       46.33         7  16.2155   340.5255
3         Health and beauty       58.22         8  23.2880   489.0480
4         Sports and travel       86.31         7  30.2085   634.3785
..                      …           …         …       …          …
995       Health and beauty       40.35         1   2.0175    42.3675
996      Home and lifestyle       97.38        10  48.6900  1022.4900
997      Food and beverages       31.84         1   1.5920    33.4320
998      Home and lifestyle       65.82         1   3.2910    69.1110
999     Fashion accessories       88.34         7  30.9190   649.2990

           date      time payment_method    cogs  gross_margin_percentage  \
0    2019-01-05  13:08:00        Ewallet  522.83                 4.761905
1    2019-03-08  10:29:00           Cash   76.40                 4.761905
2    2019-03-03  13:23:00    Credit card  324.31                 4.761905
3    2019-01-27  20:33:00        Ewallet  465.76                 4.761905
4    2019-02-08  10:37:00        Ewallet  604.17                 4.761905
..          …         …              …       …                        …
995  2019-01-29  13:46:00        Ewallet   40.35                 4.761905
996  2019-03-02  17:16:00        Ewallet  973.80                 4.761905
997  2019-02-09  13:22:00           Cash   31.84                 4.761905
998  2019-02-22  15:33:00           Cash   65.82                 4.761905
999  2019-02-18  13:28:00           Cash  618.38                 4.761905

     gross_income  rating  timeofday
0         26.1415     9.1  Afternoon
1          3.8200     9.6    Morning
2         16.2155     7.4  Afternoon
3         23.2880     8.4    Evening
4         30.2085     5.3    Morning
..              …       …          …
995        2.0175     6.2  Afternoon
996       48.6900     4.4  Afternoon
997        1.5920     7.7  Afternoon
998        3.2910     4.1  Afternoon
999       30.9190     6.6  Afternoon

[1000 rows x 18 columns]
```

[8]:
```
# Adding DAYNAME column
query1 = 'ALTER TABLE amazon ADD COLUMN dayname VARCHAR(20)'
cur.execute(query1)
query2 = 'SELECT * FROM amazon'
cur.execute(query2)
out1=cur.fetchall()
df1 = pd.DataFrame(out1, columns=['invoice_id', 'branch',␣
 ↪'city','customer_type',
'gender','product_line','unit_price','quantity','VAT','total','date','time','payment_method','
,'gross_margin_percentage','gross_income','rating','timeofday','dayname'])
df1
```

[8]:
```
      invoice_id branch       city customer_type  gender  \
0    750-67-8428      A     Yangon        Member  Female
1    226-31-3081      C  Naypyitaw        Normal  Female
2    631-41-3108      A     Yangon        Normal    Male
3    123-19-1176      A     Yangon        Member    Male
4    373-73-7910      A     Yangon        Normal    Male
..           …    …        …           …     …
995  233-67-5758      C  Naypyitaw        Normal    Male
996  303-96-2227      B   Mandalay        Normal  Female
997  727-02-1313      A     Yangon        Member    Male
998  347-56-2442      A     Yangon        Normal    Male
999  849-09-3807      A     Yangon        Member  Female

                product_line  unit_price  quantity      VAT      total  \
0          Health and beauty       74.69         7  26.1415   548.9715
1     Electronic accessories       15.28         5   3.8200    80.2200
2         Home and lifestyle       46.33         7  16.2155   340.5255
3          Health and beauty       58.22         8  23.2880   489.0480
4          Sports and travel       86.31         7  30.2085   634.3785
..                       …         …       …      …        …
995        Health and beauty       40.35         1   2.0175    42.3675
996       Home and lifestyle       97.38        10  48.6900  1022.4900
997      Food and beverages        31.84         1   1.5920    33.4320
998       Home and lifestyle       65.82         1   3.2910    69.1110
999     Fashion accessories       88.34         7  30.9190   649.2990

           date      time payment_method    cogs  gross_margin_percentage  \
0    2019-01-05  13:08:00        Ewallet  522.83                 4.761905
1    2019-03-08  10:29:00           Cash   76.40                 4.761905
2    2019-03-03  13:23:00    Credit card  324.31                 4.761905
3    2019-01-27  20:33:00        Ewallet  465.76                 4.761905
4    2019-02-08  10:37:00        Ewallet  604.17                 4.761905
..          …       …            …     …                    …
995  2019-01-29  13:46:00        Ewallet   40.35                 4.761905
996  2019-03-02  17:16:00        Ewallet  973.80                 4.761905
```

```
997  2019-02-09  13:22:00           Cash   31.84                 4.761905
998  2019-02-22  15:33:00           Cash   65.82                 4.761905
999  2019-02-18  13:28:00           Cash  618.38                 4.761905

     gross_income  rating  timeofday dayname
0         26.1415     9.1  Afternoon    None
1          3.8200     9.6    Morning    None
2         16.2155     7.4  Afternoon    None
3         23.2880     8.4    Evening    None
4         30.2085     5.3    Morning    None
..            …       …        …      …
995        2.0175     6.2  Afternoon    None
996       48.6900     4.4  Afternoon    None
997        1.5920     7.7  Afternoon    None
998        3.2910     4.1  Afternoon    None
999       30.9190     6.6  Afternoon    None

[1000 rows x 19 columns]
```

[9]:
```
query1="""UPDATE amazon
SET dayname = DATE_FORMAT(date, '%a')"""
cur.execute(query1)
query2 = 'SELECT * FROM amazon'
cur.execute(query2)
out2=cur.fetchall()
df1 = pd.DataFrame(out2,columns=['invoice_id', 'branch', 'city','customer_type',
'gender','product_line','unit_price','quantity','VAT','total','date','time','payment_method','
,'gross_margin_percentage','gross_income','rating','timeofday','dayname'])
df1
```

[9]:
```
      invoice_id branch       city customer_type  gender  \
0    750-67-8428      A     Yangon        Member  Female
1    226-31-3081      C  Naypyitaw        Normal  Female
2    631-41-3108      A     Yangon        Normal    Male
3    123-19-1176      A     Yangon        Member    Male
4    373-73-7910      A     Yangon        Normal    Male
..           …    …        …           …     …
995  233-67-5758      C  Naypyitaw        Normal    Male
996  303-96-2227      B   Mandalay        Normal  Female
997  727-02-1313      A     Yangon        Member    Male
998  347-56-2442      A     Yangon        Normal    Male
999  849-09-3807      A     Yangon        Member  Female

               product_line  unit_price  quantity      VAT      total  \
0         Health and beauty       74.69         7  26.1415   548.9715
1    Electronic accessories       15.28         5   3.8200    80.2200
2        Home and lifestyle       46.33         7  16.2155   340.5255
```

```
3          Health and beauty       58.22          8  23.2880   489.0480
4          Sports and travel       86.31          7  30.2085   634.3785
..                       …          …        …       …          …
995        Health and beauty       40.35          1   2.0175    42.3675
996       Home and lifestyle       97.38         10  48.6900  1022.4900
997      Food and beverages        31.84          1   1.5920    33.4320
998       Home and lifestyle       65.82          1   3.2910    69.1110
999     Fashion accessories        88.34          7  30.9190   649.2990

           date      time payment_method    cogs  gross_margin_percentage  \
0    2019-01-05  13:08:00        Ewallet  522.83                 4.761905
1    2019-03-08  10:29:00           Cash   76.40                 4.761905
2    2019-03-03  13:23:00    Credit card  324.31                 4.761905
3    2019-01-27  20:33:00        Ewallet  465.76                 4.761905
4    2019-02-08  10:37:00        Ewallet  604.17                 4.761905
..          …         …              …     …                      …
995  2019-01-29  13:46:00        Ewallet   40.35                 4.761905
996  2019-03-02  17:16:00        Ewallet  973.80                 4.761905
997  2019-02-09  13:22:00           Cash   31.84                 4.761905
998  2019-02-22  15:33:00           Cash   65.82                 4.761905
999  2019-02-18  13:28:00           Cash  618.38                 4.761905

     gross_income  rating  timeofday dayname
0         26.1415     9.1  Afternoon     Sat
1          3.8200     9.6    Morning     Fri
2         16.2155     7.4  Afternoon     Sun
3         23.2880     8.4    Evening     Sun
4         30.2085     5.3    Morning     Fri
..            …      …        …      …
995        2.0175     6.2  Afternoon     Tue
996       48.6900     4.4  Afternoon     Sat
997        1.5920     7.7  Afternoon     Sat
998        3.2910     4.1  Afternoon     Fri
999       30.9190     6.6  Afternoon     Mon

[1000 rows x 19 columns]
```

[10]:
```
# Adding MONTHNAME column
query1 = 'ALTER TABLE amazon ADD COLUMN monthname VARCHAR(20)'
cur.execute(query1)
query2 = 'SELECT * FROM amazon'
cur.execute(query2)
out1=cur.fetchall()
df1 = pd.DataFrame(out1, columns=['invoice_id', 'branch',␣
 ↪'city','customer_type',
'gender','product_line','unit_price','quantity','VAT','total','date','time','payment_method','
'gross_margin_percentage','gross_income','rating','timeofday','dayname','monthname'])
```

```
df1
```

```
[10]:      invoice_id branch       city customer_type  gender  \
     0    750-67-8428      A     Yangon        Member  Female
     1    226-31-3081      C  Naypyitaw        Normal  Female
     2    631-41-3108      A     Yangon        Normal    Male
     3    123-19-1176      A     Yangon        Member    Male
     4    373-73-7910      A     Yangon        Normal    Male
     ..           …    …        …           …       …
     995  233-67-5758      C  Naypyitaw        Normal    Male
     996  303-96-2227      B   Mandalay        Normal  Female
     997  727-02-1313      A     Yangon        Member    Male
     998  347-56-2442      A     Yangon        Normal    Male
     999  849-09-3807      A     Yangon        Member  Female

                    product_line  unit_price  quantity      VAT      total  \
     0         Health and beauty       74.69         7  26.1415   548.9715
     1    Electronic accessories       15.28         5   3.8200    80.2200
     2        Home and lifestyle       46.33         7  16.2155   340.5255
     3         Health and beauty       58.22         8  23.2880   489.0480
     4         Sports and travel       86.31         7  30.2085   634.3785
     ..                      …           …       …       …         …
     995       Health and beauty       40.35         1   2.0175    42.3675
     996      Home and lifestyle       97.38        10  48.6900  1022.4900
     997      Food and beverages       31.84         1   1.5920    33.4320
     998      Home and lifestyle       65.82         1   3.2910    69.1110
     999     Fashion accessories       88.34         7  30.9190   649.2990

                date      time payment_method    cogs  gross_margin_percentage  \
     0    2019-01-05  13:08:00        Ewallet  522.83                 4.761905
     1    2019-03-08  10:29:00           Cash   76.40                 4.761905
     2    2019-03-03  13:23:00    Credit card  324.31                 4.761905
     3    2019-01-27  20:33:00        Ewallet  465.76                 4.761905
     4    2019-02-08  10:37:00        Ewallet  604.17                 4.761905
     ..          …       …            …     …                      …
     995  2019-01-29  13:46:00        Ewallet   40.35                 4.761905
     996  2019-03-02  17:16:00        Ewallet  973.80                 4.761905
     997  2019-02-09  13:22:00           Cash   31.84                 4.761905
     998  2019-02-22  15:33:00           Cash   65.82                 4.761905
     999  2019-02-18  13:28:00           Cash  618.38                 4.761905

          gross_income  rating  timeofday dayname monthname
     0         26.1415     9.1  Afternoon     Sat      None
     1          3.8200     9.6    Morning     Fri      None
     2         16.2155     7.4  Afternoon     Sun      None
     3         23.2880     8.4    Evening     Sun      None
     4         30.2085     5.3    Morning     Fri      None
```

```
..            …      …        …       …       …
995         2.0175     6.2  Afternoon     Tue       None
996        48.6900     4.4  Afternoon     Sat       None
997         1.5920     7.7  Afternoon     Sat       None
998         3.2910     4.1  Afternoon     Fri       None
999        30.9190     6.6  Afternoon     Mon       None

[1000 rows x 20 columns]
```

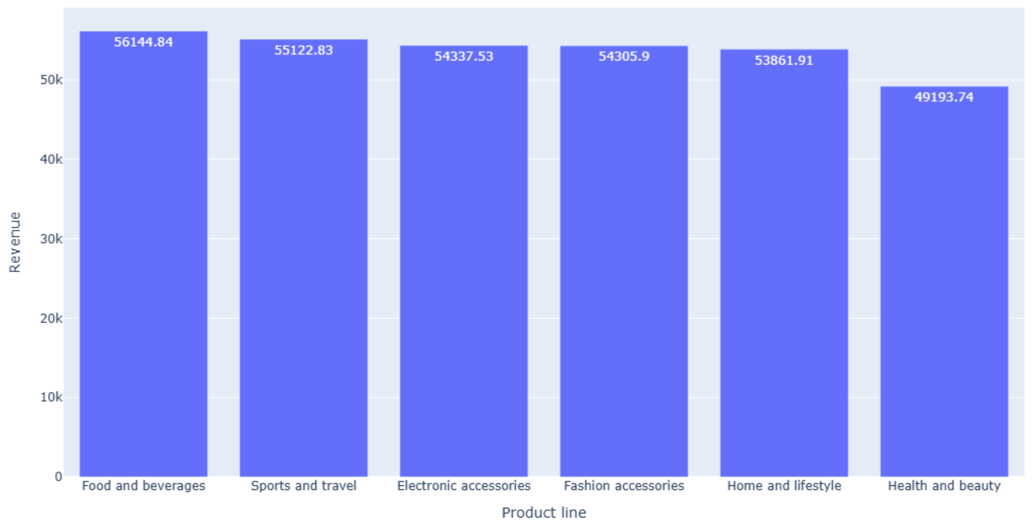
[11]:
```
query1="""UPDATE amazon
SET monthname = DATE_FORMAT(date, '%b')"""
cur.execute(query1)
query2 = 'SELECT * FROM amazon'
cur.execute(query2)
out2=cur.fetchall()
df1 = pd.DataFrame(out2,columns=['invoice_id', 'branch', 'city','customer_type',
'gender','product_line','unit_price','quantity','VAT','total','date','time','payment_method','
'gross_margin_percentage','gross_income','rating','timeofday','dayname','monthname'])
df1
```

[11]:
```
      invoice_id branch       city customer_type  gender  \
0    750-67-8428      A     Yangon        Member  Female
1    226-31-3081      C  Naypyitaw        Normal  Female
2    631-41-3108      A     Yangon        Normal    Male
3    123-19-1176      A     Yangon        Member    Male
4    373-73-7910      A     Yangon        Normal    Male
..           …    …        …           …     …
995  233-67-5758      C  Naypyitaw        Normal    Male
996  303-96-2227      B   Mandalay        Normal  Female
997  727-02-1313      A     Yangon        Member    Male
998  347-56-2442      A     Yangon        Normal    Male
999  849-09-3807      A     Yangon        Member  Female

              product_line  unit_price  quantity      VAT      total  \
0        Health and beauty       74.69         7  26.1415   548.9715
1   Electronic accessories       15.28         5   3.8200    80.2200
2       Home and lifestyle       46.33         7  16.2155   340.5255
3        Health and beauty       58.22         8  23.2880   489.0480
4        Sports and travel       86.31         7  30.2085   634.3785
..                     …         …       …      …        …
995      Health and beauty       40.35         1   2.0175    42.3675
996     Home and lifestyle       97.38        10  48.6900  1022.4900
997     Food and beverages       31.84         1   1.5920    33.4320
998     Home and lifestyle       65.82         1   3.2910    69.1110
999    Fashion accessories       88.34         7  30.9190   649.2990

           date      time payment_method    cogs  gross_margin_percentage  \
0    2019-01-05  13:08:00       Ewallet  522.83                 4.761905
1    2019-03-08  10:29:00          Cash   76.40                 4.761905
2    2019-03-03  13:23:00   Credit card  324.31                 4.761905
3    2019-01-27  20:33:00       Ewallet  465.76                 4.761905
4    2019-02-08  10:37:00       Ewallet  604.17                 4.761905
..          …         …             …       …                        …
995  2019-01-29  13:46:00       Ewallet   40.35                 4.761905
996  2019-03-02  17:16:00       Ewallet  973.80                 4.761905
997  2019-02-09  13:22:00          Cash   31.84                 4.761905
998  2019-02-22  15:33:00          Cash   65.82                 4.761905
999  2019-02-18  13:28:00          Cash  618.38                 4.761905

     gross_income  rating  timeofday dayname monthname
0         26.1415     9.1  Afternoon     Sat       Jan
1          3.8200     9.6    Morning     Fri       Mar
2         16.2155     7.4  Afternoon     Sun       Mar
3         23.2880     8.4    Evening     Sun       Jan
4         30.2085     5.3    Morning     Fri       Feb
..            …       …         …       …         …
995        2.0175     6.2  Afternoon     Tue       Jan
996       48.6900     4.4  Afternoon     Sat       Mar
997        1.5920     7.7  Afternoon     Sat       Feb
998        3.2910     4.1  Afternoon     Fri       Feb
999       30.9190     6.6  Afternoon     Mon       Feb

[1000 rows x 20 columns]
```

Product__Analysis

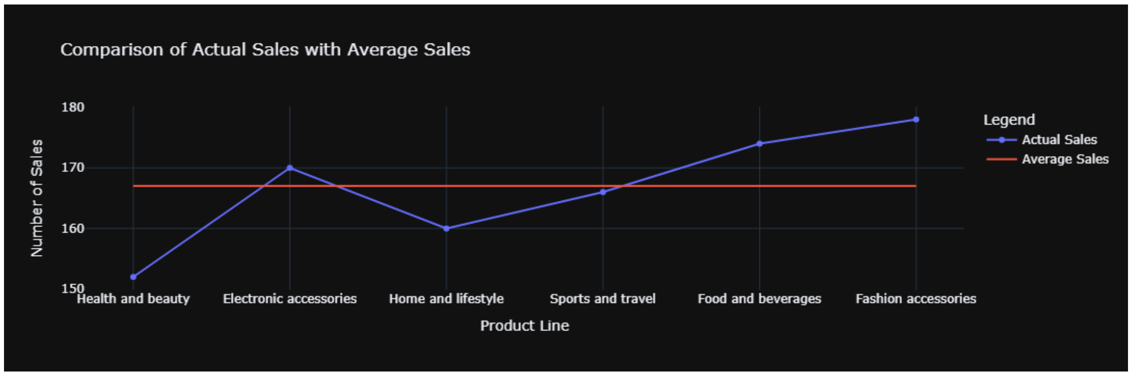
[13]:
```
1# Conduct analysis on the data to understand the different product lines,
# the products lines performing best and the product lines that need to be␣
↪improved.
query1="""SELECT `Product line`,ROUND(sum(total),2) as Total_Revenue,(SELECT␣
↪ROUND(SUM(`total`)/COUNT(DISTINCT `Product line`),2) from amazon) AS␣
↪Average_Revenue,
RANK() OVER(ORDER BY ROUND(sum(total),2) DESC) AS Revenue_Rank FROM amazon␣
↪group by `Product line` ORDER BY Total_Revenue DESC"""
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Product␣
↪line','Revenue','Average_Revenue','Rank'])
print(df1)

fig = px.bar(df1, x="Product line", y="Revenue",text='Revenue',
             height=600)
fig.show()
```

```
              Product line   Revenue  Average_Revenue  Rank
0      Food and beverages  56144.84         53827.79     1
1        Sports and travel  55122.83         53827.79     2
2   Electronic accessories  54337.53         53827.79     3
3      Fashion accessories  54305.90         53827.79     4
4       Home and lifestyle  53861.91         53827.79     5
5        Health and beauty  49193.74         53827.79     6
```



```
[14]: #Product line-Comparision of  Sales
query1="""SELECT `Product line`,count(`Invoice id`) as Sales,
(SELECT ROUND(COUNT(`Invoice id`)/COUNT(DISTINCT `Product line`),0) FROM␣
 ↪amazon)  as Average_Sales FROM amazon group by `Product line`"""
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Product line','Sales','Average_Sales'])
print(df1)

fig = go.Figure()
fig.add_trace(go.Scatter(x=df1['Product line'], y=df1['Sales'],␣
 ↪mode='markers+lines', name='Actual Sales'))
fig.add_trace(go.Scatter(x=df1['Product line'], y=df1['Average_Sales'],␣
 ↪mode='lines', name='Average Sales'))
fig.update_layout(
    title='Comparison of Actual Sales with Average Sales',
    xaxis_title='Product Line',
    yaxis_title='Number of Sales',
    legend_title='Legend',
    template='plotly_dark'
)

# Show the plot
fig.show()
```

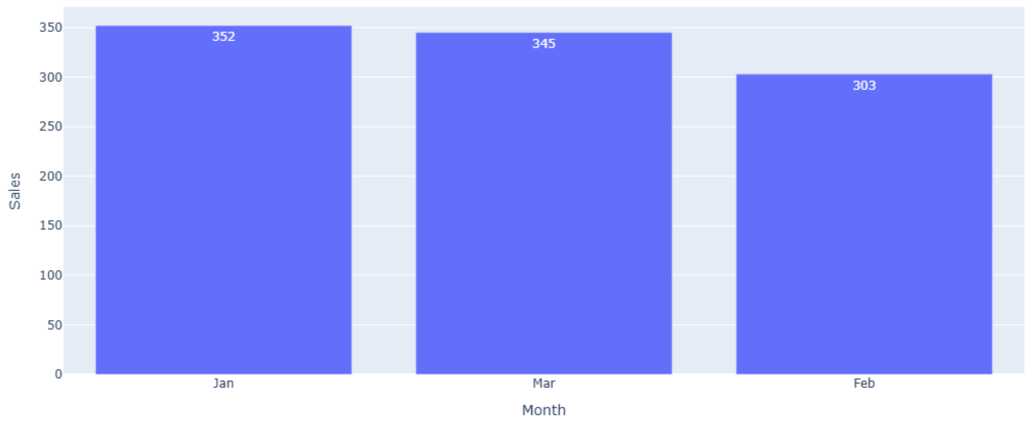
```
             Product line  Sales  Average_Sales
0        Health and beauty    152            167
1   Electronic accessories    170            167
2       Home and lifestyle    160            167
3        Sports and travel    166            167
4       Food and beverages    174            167
5      Fashion accessories    178            167
```



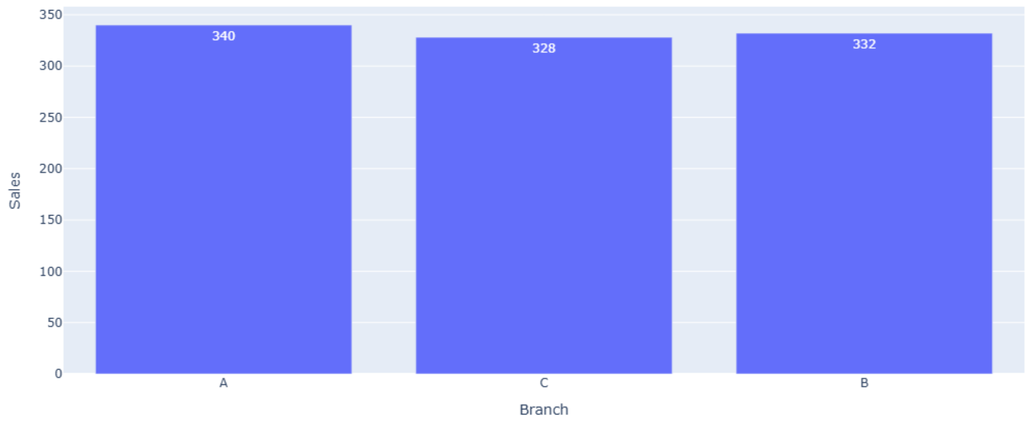
Sales_Analysis

[35]:
```
# Sales BY Month
query1=" SELECT DATE_FORMAT(date, '%b') as Month,COUNT(`Invoice id`) AS Sales␣
 ↪FROM amazon GROUP BY Month"
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Month','Sales'])
print(df1)

fig = px.bar(df1, x="Month", y="Sales", barmode='group',text='Sales',
             height=500,width=600)
fig.show()
```

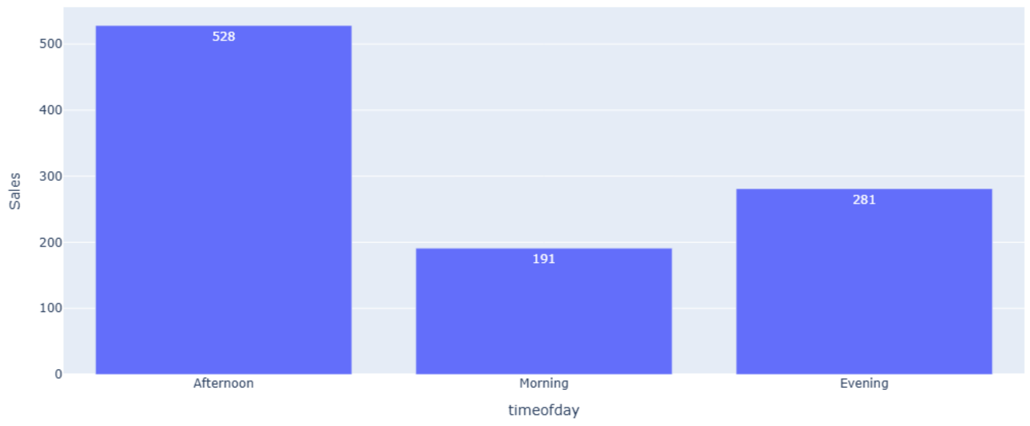
```
  Month  Sales
0   Jan    352
1   Mar    345
2   Feb    303
```



[37]:
```
# Branch Performance in Sales
query1='SELECT Branch,count(`Invoice id`) as Sales FROM amazon group by Branch'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Branch','Sales'])
df1

fig = px.bar(df1, x="Branch", y="Sales", barmode='group',text='Sales',
             height=500,width=500)
fig.show()
```



```
[39]: # Sales-Timeofday
query1='SELECT timeofday,count(`Invoice id`) as Sales FROM amazon group by␣
  ↪timeofday'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['timeofday','Sales'])
df1

fig = px.bar(df1, x="timeofday", y="Sales", barmode='group',text='Sales',
             height=500,width=500)
fig.show()
```



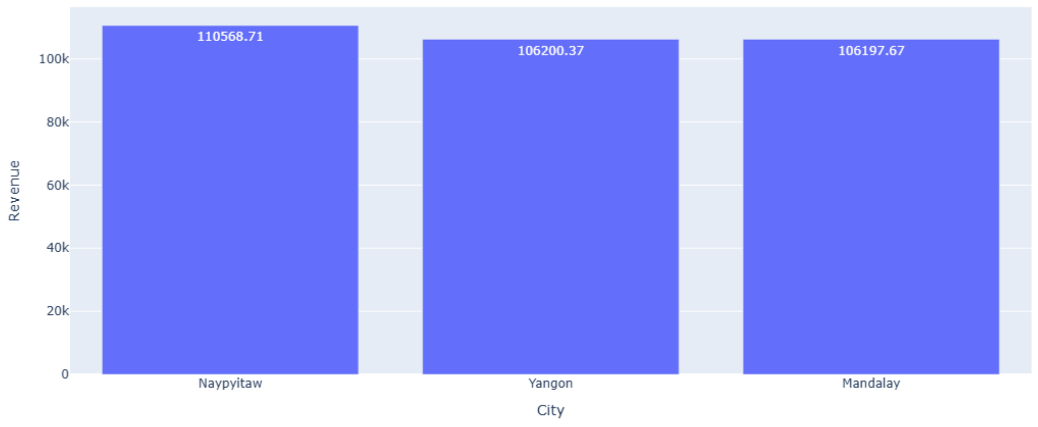
```
[41]: #Sales-Dayofweek
query1="""
with SalesDays as (SELECT branch,dayname,COUNT(`Invoice ID`) AS Sales FROM␣
  ↪amazon GROUP BY branch,dayname ORDER BY BRANCH),
MaxSales as (SELECT branch,MAX(Sales) AS MaxSales FROM SalesDays GROUP BY␣
  ↪branch)
SELECT sd.branch,sd.dayname,Sales FROM SalesDays sd JOIN MaxSales ms ON ms.
  ↪branch=sd.branch AND ms.MaxSales=sd.Sales
"""
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Branch','dayname','Sales'])
df1
```

```
[41]:   Branch dayname  Sales
      0      A     Sun     52
      1      B     Sat     60
```

```
2       C      Sat      54
3       C      Tue      54
```

[43]:
```
# City contribution
query1='SELECT City,Round(SUM(Total),2) as Revenue FROM amazon group by City␣
  ↪ORDER BY Revenue DESC'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['City','Revenue'])
print(df1)

fig = px.bar(df1, x="City", y="Revenue", barmode='group',text='Revenue',
             height=500,width=600)
fig.show()
```

```
        City    Revenue
0  Naypyitaw  110568.71
1     Yangon  106200.37
2   Mandalay  106197.67
```

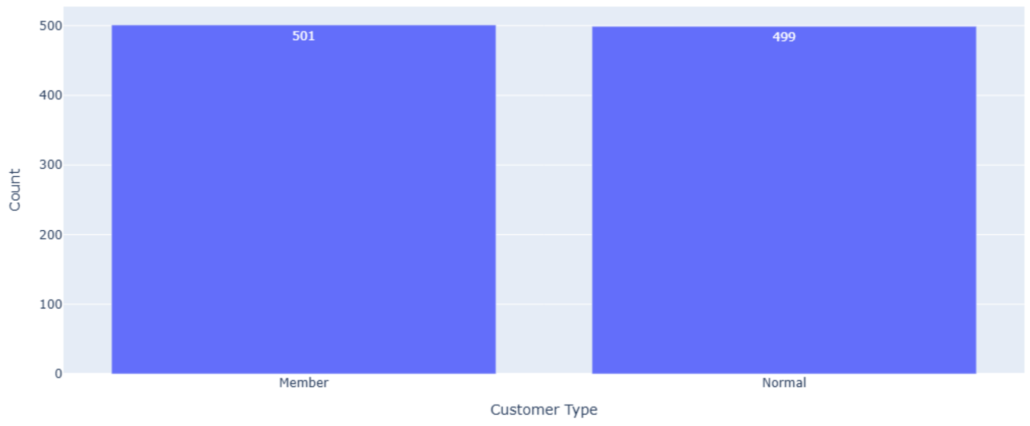


Customer Analysis

[45]:
```
#Customer segments
#customer types
query1='SELECT `Customer Type`,COUNT(`Customer Type`) AS Count FROM amazon␣
  ↪GROUP BY `Customer Type` ORDER BY Count DESC'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Customer Type','Count'])
print(df1)
```

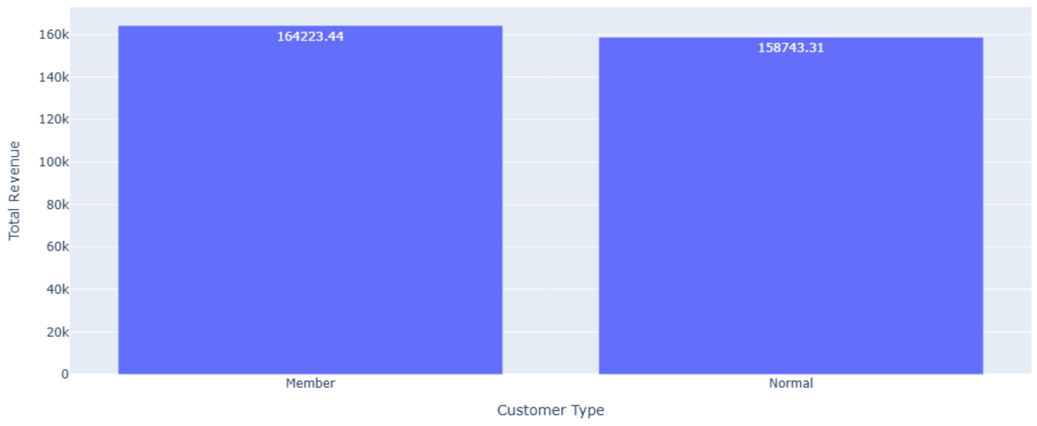
```
fig = px.bar(df1, x="Customer Type", y="Count", barmode='group',text='Count',
             height=500,width=400)
fig.show()
```

```
  Customer Type  Count
0        Member    501
1        Normal    499
```



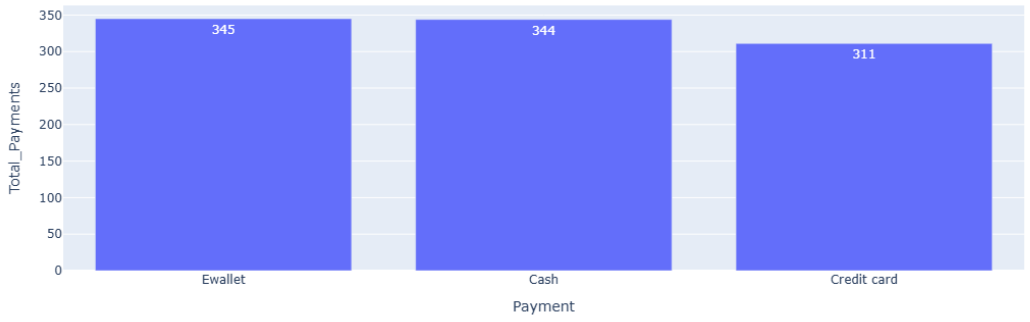
```
[47]: # customer type contributing the highest revenue
query1='SELECT `Customer Type`,ROUND(SUM(Total),2) AS Count FROM amazon GROUP␣
 ↪BY `Customer Type` ORDER BY Count DESC'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Customer Type','Total Revenue'])
print(df1)
fig = px.bar(df1, x="Customer Type", y="Total Revenue",␣
 ↪barmode='group',text='Total Revenue',
             height=500,width=400)
fig.show()
```

```
  Customer Type  Total Revenue
0        Member      164223.44
1        Normal      158743.31
```



[49]:
```python
# customer payment methods
query1='SELECT Payment, COUNT(Payment) AS Total_Payments FROM amazon GROUP BY␣
↪Payment ORDER BY Total_Payments DESC'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Payment','Total_Payments'])
print(df1)
fig = px.bar(df1, x="Payment", y="Total_Payments",␣
↪barmode='group',text='Total_Payments',
             height=400,width=500)
fig.show()
```

```
       Payment  Total_Payments
0      Ewallet             345
1         Cash             344
2  Credit card             311
```



Business_Questions_to_Answer

```
[51]: # 1) What is the count of distinct cities in the dataset?
query1='SELECT COUNT(DISTINCT city) AS Total_Cities from amazon'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Total_Cities'])
df1
```

```
[51]:    Total_Cities
      0             3
```

```
[53]: # 2) For each branch, what is the corresponding city?
query1='SELECT Branch,city from amazon GROUP BY Branch,city'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Branch','City'])
df1
```

```
[53]:   Branch       City
      0      A     Yangon
      1      C  Naypyitaw
      2      B   Mandalay
```

```
[55]: # 3) What is the count of distinct product lines in the dataset?
query1='SELECT COUNT(DISTINCT `Product line`) as Total_Product_lines from␣
 ↪amazon'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Total_Product_lines'])
df1
```

```
[55]:    Total_Product_lines
      0                    6
```

```
[57]: # 4) Which payment method occurs most frequently?
query1='SELECT Payment, COUNT(Payment) AS Total_Payments FROM amazon GROUP BY␣
 ↪Payment ORDER BY Total_Payments DESC LIMIT 1'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Payment','Total_Payments'])
df1
```

```
[57]:    Payment  Total_Payments
      0  Ewallet             345
```

```
[59]: # 5) Which product line has the highest sales?
      query1='SELECT `Product line`,Count(`Invoice Id`) AS Sales FROM amazon GROUP BY␣
       ↪`Product line` ORDER BY Sales DESC LIMIT 1'
      cur.execute(query1)
      out1=cur.fetchall()
      df1 = pd.DataFrame(out1,columns=['Product line','Sales'])
      df1
```

```
[59]:           Product line  Sales
      0  Fashion accessories    178
```

```
[61]: # 6) How much revenue is generated each month?
      query1="SELECT DATE_FORMAT(date, '%b') as Month,SUM(total) AS Total_Revenue␣
       ↪FROM amazon GROUP BY Month"
      cur.execute(query1)
      out1=cur.fetchall()
      df1 = pd.DataFrame(out1,columns=['Month','Revenue'])
      df1
```

```
[61]:   Month     Revenue
      0   Jan  116291.868
      1   Mar  109455.507
      2   Feb   97219.374
```

```
[63]: # 7) In which month did the cost of goods sold reach its peak?
      query1="SELECT DATE_FORMAT(date, '%b') as Month,SUM(cogs) AS Total_cogs FROM␣
       ↪amazon GROUP BY Month ORDER BY Total_cogs DESC LIMIT 1"
      cur.execute(query1)
      out1=cur.fetchall()
      df1 = pd.DataFrame(out1,columns=['Month','Total_cogs'])
      df1
```

```
[63]:   Month  Total_cogs
      0   Jan   110754.16
```

```
[65]: # 8) Which product line generated the highest revenue?
      query1='SELECT `Product line`,sum(total) as Total_Revenue FROM amazon group by␣
       ↪`Product line` ORDER BY Total_Revenue DESC LIMIT 1'
      cur.execute(query1)
      out1=cur.fetchall()
      df1 = pd.DataFrame(out1,columns=['Month','Revenue'])
      df1
```

```
[65]:                 Month    Revenue
      0  Food and beverages  56144.844
```

[67]:
```
# 9) In which city was the highest revenue recorded?
query1='SELECT city,sum(total) as Total_Revenue FROM amazon group by city ORDER␣
 ↪BY Total_Revenue DESC LIMIT 1'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['City','Revenue'])
df1
```

[67]:
```
        City      Revenue
0  Naypyitaw  110568.7065
```

[107]:
```
# 10) Which product line incurred the highest Value Added Tax?
query1='SELECT `Product line`,max(`Tax 5%`) as Highest_Vat FROM amazon group by␣
 ↪`Product line` ORDER BY Highest_Vat DESC LIMIT 1'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['City','VAT'])
df1
```

[107]:
```
                 City    VAT
0  Fashion accessories  49.65
```

[71]:
```
# 11) For each product line, add a column indicating "Good" if its sales are␣
 ↪above average, otherwise "Bad."
query1="""
with SalesCount as (SELECT `Product line`,COUNT(`Invoice ID`) AS Sales FROM␣
 ↪amazon GROUP BY `Product line`),
AverageSales as (SELECT `Product line`,AVG(Sales) OVER () AS Average_Sales FROM␣
 ↪SalesCount)
SELECT sc.`Product line`,sc.Sales, avs.Average_Sales,
CASE
 WHEN Sales>Average_Sales THEN 'Good'
 ELSE 'Bad'
 END AS Performance
 FROM SalesCount sc JOIN AverageSales avs ON sc.`Product line`=avs.`Product␣
 ↪line`
"""
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Product line','Sales','Average␣
 ↪Sales','Performance'])
df1
```

[71]:
```
             Product line  Sales  Average Sales Performance
0       Health and beauty    152       166.6667         Bad
1  Electronic accessories    170       166.6667        Good
2      Home and lifestyle    160       166.6667         Bad
```

```
3       Sports and travel    166      166.6667         Bad
4      Food and beverages    174      166.6667        Good
5     Fashion accessories    178      166.6667        Good
```

[73]:
```
# 12) Identify the branch that exceeded the average number of products sold.
query1="""with ProductsCount as (SELECT branch,SUM(`Quantity`) AS Sales FROM␣
 ↪amazon GROUP BY branch),
AverageProductsSold as (SELECT branch,AVG(Sales) OVER () AS␣
 ↪Average_Products_Sold FROM ProductsCount)
SELECT pc.branch,pc.Sales, aps.Average_Products_Sold
FROM ProductsCount pc JOIN AverageProductsSold aps ON pc.branch=aps.branch␣
 ↪WHERE sales>Average_Products_Sold"""
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Branch','Total_Quantity','Average_Quantity'])
df1
```

[73]:
```
  Branch Total_Quantity Average_Quantity
0      A           1859        1836.6667
```

[75]:
```
# 13) Which product line is most frequently associated with each gender?
query1="""WITH ProductLineCounts AS (
    SELECT Gender,`Product line`, COUNT(*) AS Count
    FROM amazon
    GROUP BY Gender, `Product line`
),
MaxCounts AS (
    SELECT Gender, MAX(Count) AS MaxCount
    FROM ProductLineCounts
    GROUP BY Gender
)
SELECT plc.Gender, plc.`Product line`, plc.Count
FROM ProductLineCounts plc
JOIN MaxCounts mc
ON plc.Gender = mc.Gender AND plc.Count = mc.MaxCount;
"""
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Gender','Product line','Count'])
df1
```

[75]:
```
   Gender         Product line  Count
0    Male     Health and beauty     88
1  Female   Fashion accessories     96
```

[77]:
```
# 14) Calculate the average rating for each product line.
```

```
query1='SELECT `Product line`,ROUND(AVG(`Rating`),2) as Avg_Rating FROM amazon␣
  ↪group by `Product line`'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Product line','Avg_Rating'])
df1
```

[77]:
```
             Product line  Avg_Rating
0       Health and beauty        7.00
1  Electronic accessories        6.92
2      Home and lifestyle        6.84
3       Sports and travel        6.92
4      Food and beverages        7.11
5     Fashion accessories        7.03
```

[79]:
```
# 15) Count the sales occurrences for each time of day on every weekday.
query1="""select dayname,timeofday,count(*) as SalesCount from amazon where␣
  ↪dayname not in ('Sat','Sun')group by dayname,timeofday
ORDER BY FIELD(dayname, 'Mon', 'Tue', 'Wed', 'Thu', 'Fri'),␣
  ↪FIELD(timeofday,'Morning','Afternoon','Evening')"""
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['dayname','timeofday','SalesCount'])
df1
```

[79]:
```
   dayname  timeofday  SalesCount
0      Mon    Morning          21
1      Mon  Afternoon          75
2      Mon    Evening          29
3      Tue    Morning          36
4      Tue  Afternoon          71
5      Tue    Evening          51
6      Wed    Morning          22
7      Wed  Afternoon          81
8      Wed    Evening          40
9      Thu    Morning          33
10     Thu  Afternoon          76
11     Thu    Evening          29
12     Fri    Morning          29
13     Fri  Afternoon          74
14     Fri    Evening          36
```

[81]:
```
# 16) Identify the customer type contributing the highest revenue.
query1='SELECT `Customer type`,SUM(Total) as Total_Revenue FROM amazon group by␣
  ↪`Customer type` LIMIT 1'
cur.execute(query1)
out1=cur.fetchall()
```

```
df1 = pd.DataFrame(out1,columns=['Customer type','Total_Revenue'])
df1
```

[81]:
```
  Customer type  Total_Revenue
0        Member     164223.444
```

[83]:
```
# 17) Determine the city with the highest VAT percentage.
query1='SELECT City,MAX(`Tax 5%`) as VAT FROM amazon group by City ORDER BY VAT␣
 ↪DESC LIMIT 1'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['City','VAT'])
df1
```

[83]:
```
        City    VAT
0  Naypyitaw  49.65
```

[85]:
```
# 18) Identify the customer type with the highest VAT payments.
query1='SELECT `Customer type`,MAX(`Tax 5%`) as VAT FROM amazon group by␣
 ↪`Customer type` ORDER BY VAT DESC  LIMIT 1'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Customer type','VAT'])
df1
```

[85]:
```
  Customer type    VAT
0        Member  49.65
```

[87]:
```
# 19) What is the count of distinct customer types in the dataset?
query1='SELECT COUNT(DISTINCT `Customer type`) AS Total_Customer_Types FROM␣
 ↪amazon'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=[' Total_Customer_Types'])
df1
```

[87]:
```
    Total_Customer_Types
0                      2
```

[89]:
```
# 20) What is the count of distinct payment methods in the dataset?
query1='SELECT COUNT(DISTINCT Payment) AS Total_Payment_Methods FROM amazon'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Total_Payment_Methods'])
df1
```

[89]:
```
   Total_Payment_Methods
0                      3
```

[91]:
```
# 21) Which customer type occurs most frequently?
query1='SELECT `Customer Type`,COUNT(`Customer Type`) AS Count FROM amazon␣
 ↪GROUP BY `Customer Type` ORDER BY Count DESC LIMIT 1'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Customer Type','Count'])
df1
```

[91]:
```
  Customer Type  Count
0        Member    501
```

[93]:
```
# 22) Identify the customer type with the highest purchase frequency.
query1='SELECT `Customer Type`,COUNT(`Invoice id`) AS Count FROM amazon GROUP␣
 ↪BY `Customer Type` ORDER BY Count DESC LIMIT 1'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Customer Type','Purchase Frequency'])
df1
```

[93]:
```
  Customer Type  Purchase Frequency
0        Member                 501
```

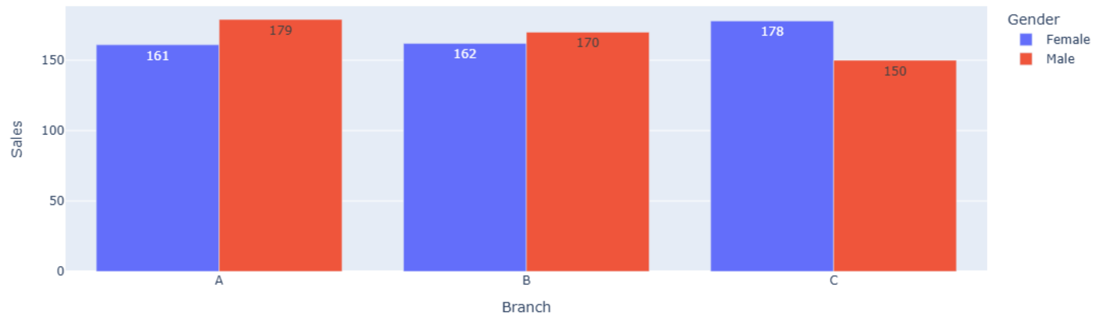
[95]:
```
# 23) Determine the predominant gender among customers.
query1='SELECT Gender,COUNT(Gender) AS Count FROM amazon GROUP BY Gender ORDER␣
 ↪BY Count DESC LIMIT 1'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Gender','Sales'])
df1
```

[95]:
```
   Gender  Sales
0  Female    501
```

[97]:
```
# 24) Examine the distribution of genders within each branch.
query1='SELECT Branch,Gender,COUNT(Gender) AS Count FROM amazon GROUP BY␣
 ↪Branch,Gender ORDER BY BRANCH'
cur.execute(query1)
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Branch','Gender','Sales'])
print(df1)

fig = px.bar(df1, x="Branch", y="Sales",
             color='Gender', barmode='group',
             height=400,text='Sales')
```

```
fig.show()
```

```
  Branch  Gender  Sales
0      A  Female    161
1      A    Male    179
2      B  Female    162
3      B    Male    170
4      C  Female    178
5      C    Male    150
```



```
[111]: # 25)Identify the time of day when customers provide the most ratings.
       query1='SELECT timeofday,COUNT(rating) AS Total_Ratings FROM amazon GROUP BY␣
        ↪timeofday ORDER BY Total_Ratings DESC LIMIT 1'
       cur.execute(query1)
       out1=cur.fetchall()
       df1 = pd.DataFrame(out1,columns=['timeofday','Total_Ratings'])
       df1
```

```
[111]:    timeofday  Total_Ratings
       0  Afternoon            528
```

```
[117]: # 26) Determine the time of day with the highest customer ratings for each␣
        ↪branch.
       query1="""
       with Ratings as (SELECT branch,timeofday,ROUND(avg(rating),2) AS␣
        ↪Average_Ratings FROM amazon GROUP BY branch,timeofday order by branch),
       MaxRatings as (SELECT branch,max(Average_Ratings) AS maxratings FROM Ratings␣
        ↪GROUP BY branch)
       select r.branch,r.timeofday,r.Average_Ratings from Ratings r join MaxRatings mr␣
        ↪on mr.branch=r.branch and r.Average_Ratings=mr.maxratings
       """
       cur.execute(query1)
```

```
out1=cur.fetchall()
df1 = pd.DataFrame(out1,columns=['Branch','Timeofday','Average_Ratings'])
df1
```

```
[117]:   Branch  Timeofday  Average_Ratings
       0      A  Afternoon             7.06
       1      B    Morning             6.89
       2      C  Afternoon             7.10
```

```
[103]: # 27) Identify the day of the week with the highest average ratings.
       query1='SELECT dayname,avg(rating) AS Average_Ratings FROM amazon GROUP BY␣
        ↪dayname ORDER BY Average_Ratings DESC LIMIT 1'
       cur.execute(query1)
       out1=cur.fetchall()
       df1 = pd.DataFrame(out1,columns=['dayname','Average_Ratings'])
       df1
```

```
[103]:   dayname  Average_Ratings
       0     Mon           7.1536
```

```
[105]: # 28) Determine the day of the week with the highest average ratings for each␣
        ↪branch.
       query1="""
       with AverageCounts as (SELECT dayname,BRANCH,ROUND(avg(rating),2) AS␣
        ↪Average_Ratings FROM amazon GROUP BY BRANCH,dayname ORDER BY branch,
       field(dayname,'Sun','Mon','Tue','Wed','Thu','Fri','Sat'))
       ,MaxCounts as (SELECT BRANCH,MAX(Average_Ratings) AS Max_Ratings FROM␣
        ↪AverageCounts GROUP BY BRANCH ORDER BY branch)
       SELECT ac.dayname,ac.BRANCH,Average_Ratings FROM AverageCounts ac JOIN␣
        ↪MaxCounts mc ON mc.Max_Ratings = ac.Average_Ratings AND mc.branch=ac.
        ↪branch"""
       cur.execute(query1)
       out1=cur.fetchall()
       df1 = pd.DataFrame(out1,columns=['dayname','Branch','Average_Ratings'])
       df1
```

```
[105]:   dayname Branch  Average_Ratings
       0     Fri      A             7.31
       1     Mon      B             7.34
       2     Fri      C             7.28
```