Importing the Necessary Libraries

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from xgboost import XGBRegressor
```

Importing the Data

```
train = pd.read_csv("/content/train.csv")
test = pd.read_csv("/content/test_QoiMO9B.csv")
```

```
train
```

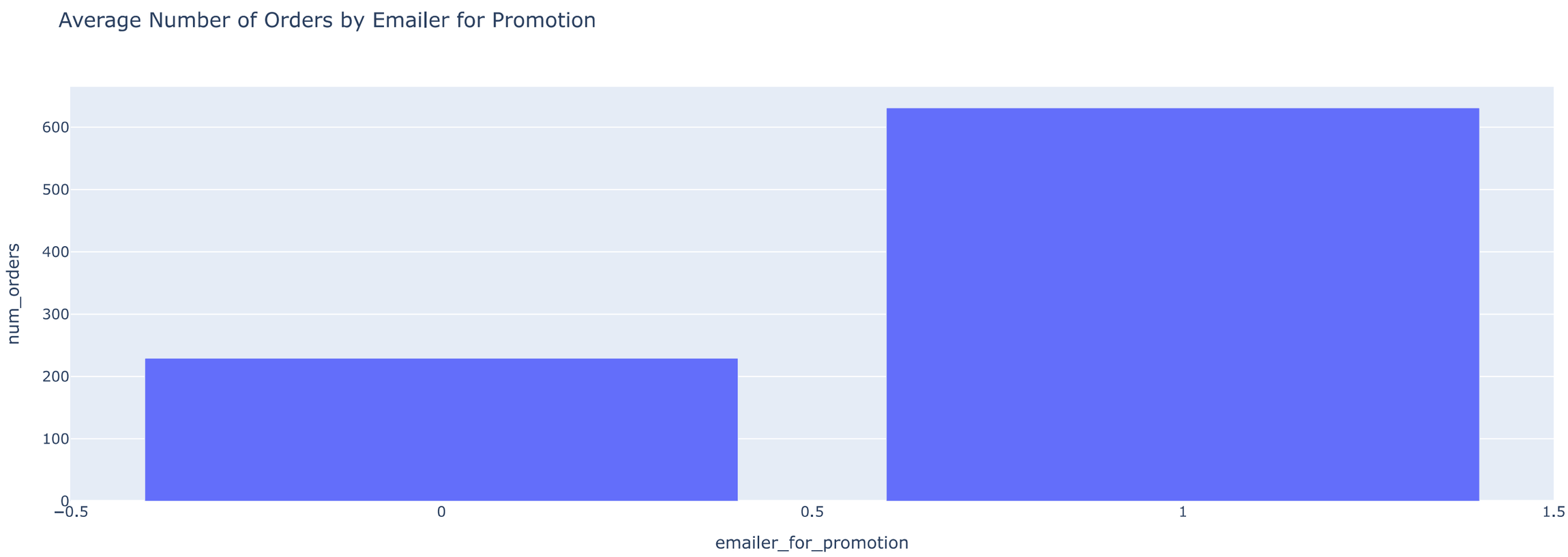
	id	week	center_id	meal_id	checkout_price	base_price	emailer_for_promotion	homepage_featured	num_orders
0	1379560	1	55	1885	136.83	152.29	0	0	177
1	1466964	1	55	1993	136.83	135.83	0	0	270
2	1346989	1	55	2539	134.86	135.86	0	0	189
3	1338232	1	55	2139	339.50	437.53	0	0	54
4	1448490	1	55	2631	243.50	242.50	0	0	40
...	...	...	...	...	...	...	...	...	...
456543	1271326	145	61	1543	484.09	484.09	0	0	68
456544	1062036	145	61	2304	482.09	482.09	0	0	42
456545	1110849	145	61	2664	237.68	321.07	0	0	501
456546	1147725	145	61	2569	243.50	313.34	0	0	729
456547	1361984	145	61	2490	292.03	290.03	0	0	162

456548 rows × 9 columns

```
train.dropna(inplace=True)
```

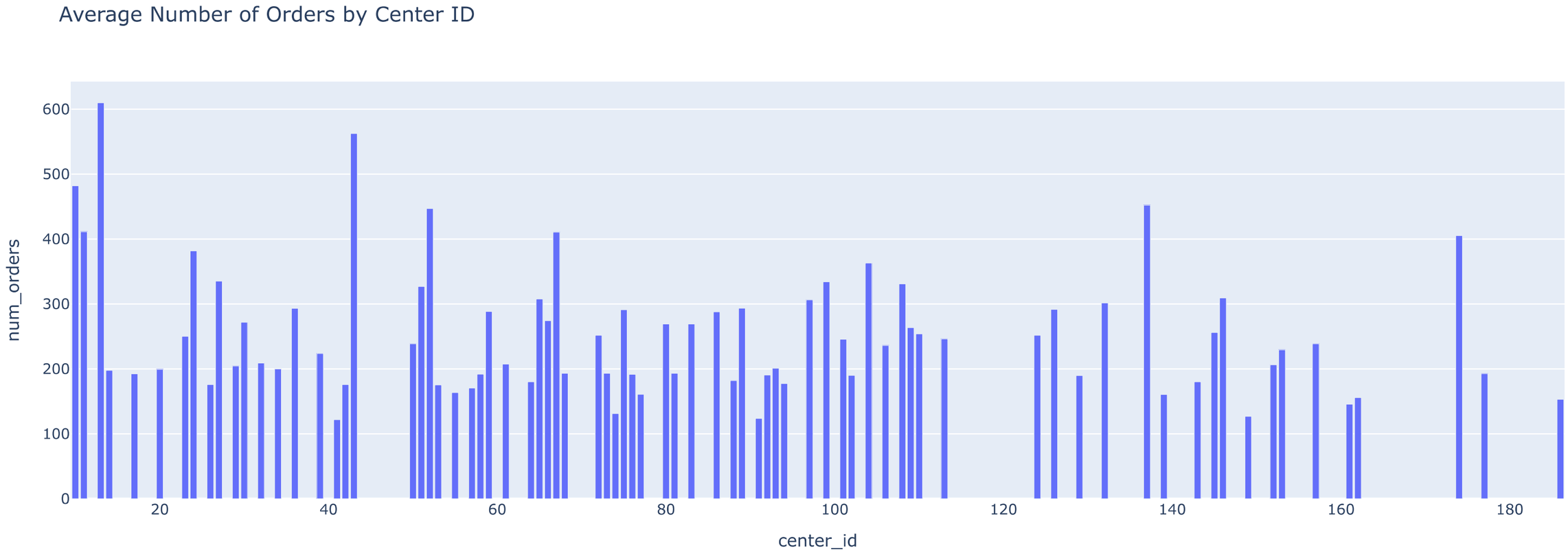
Plotting the average the number of Orders by Emailers(for promotion)

```
import plotly.express as px
px.bar(train.groupby('emailer_for_promotion')['num_orders'].mean().reset_index(),
       x='emailer_for_promotion',
       y='num_orders',
       title='Average Number of Orders by Emailer for Promotion'
)
```



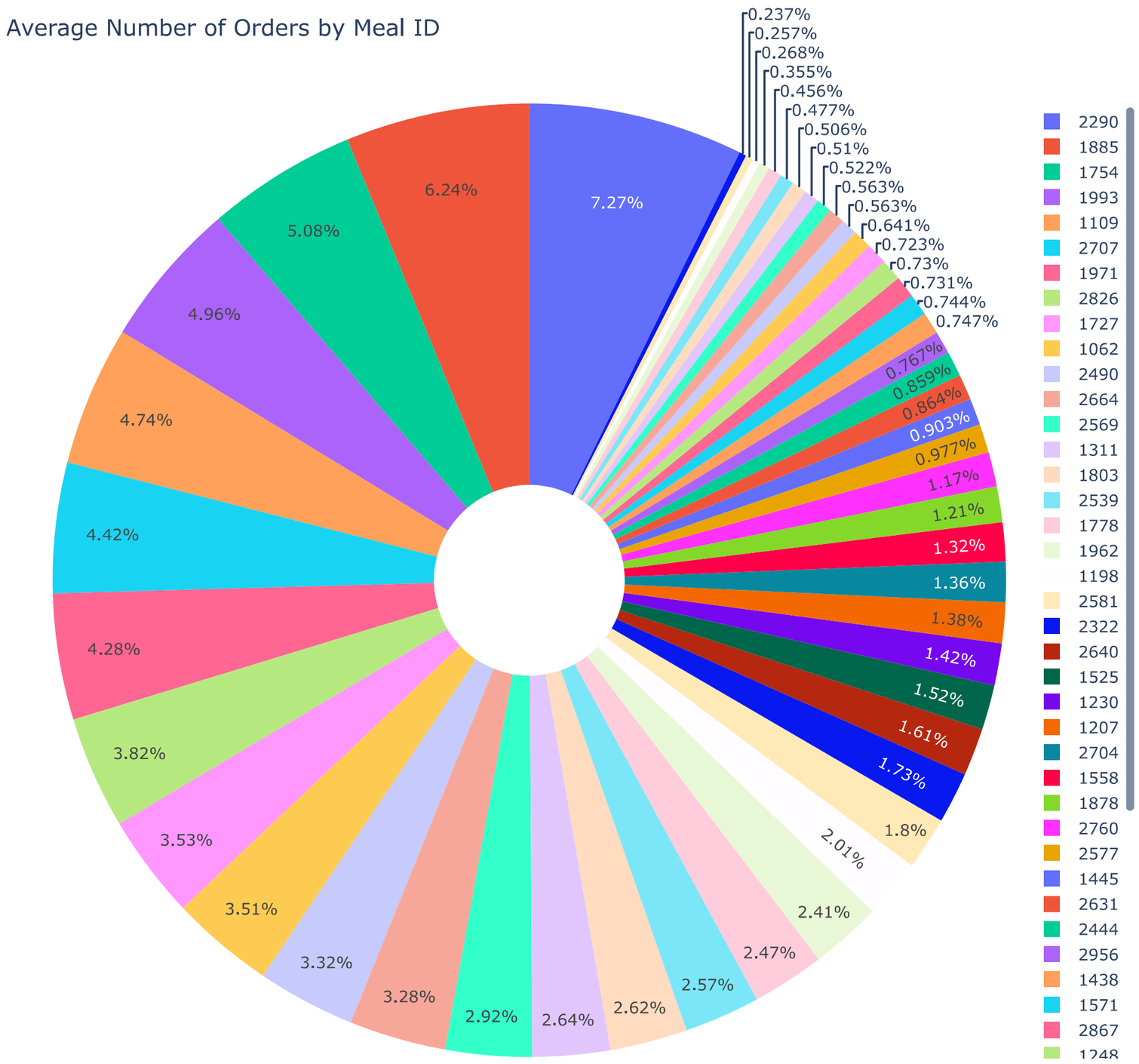
Plottign the Average Number of Orders by Center ID

```
px.bar(
    train.groupby('center_id')['num_orders'].mean().reset_index(),
    x='center_id',
    y='num_orders',
    title='Average Number of Orders by Center ID'
)
```



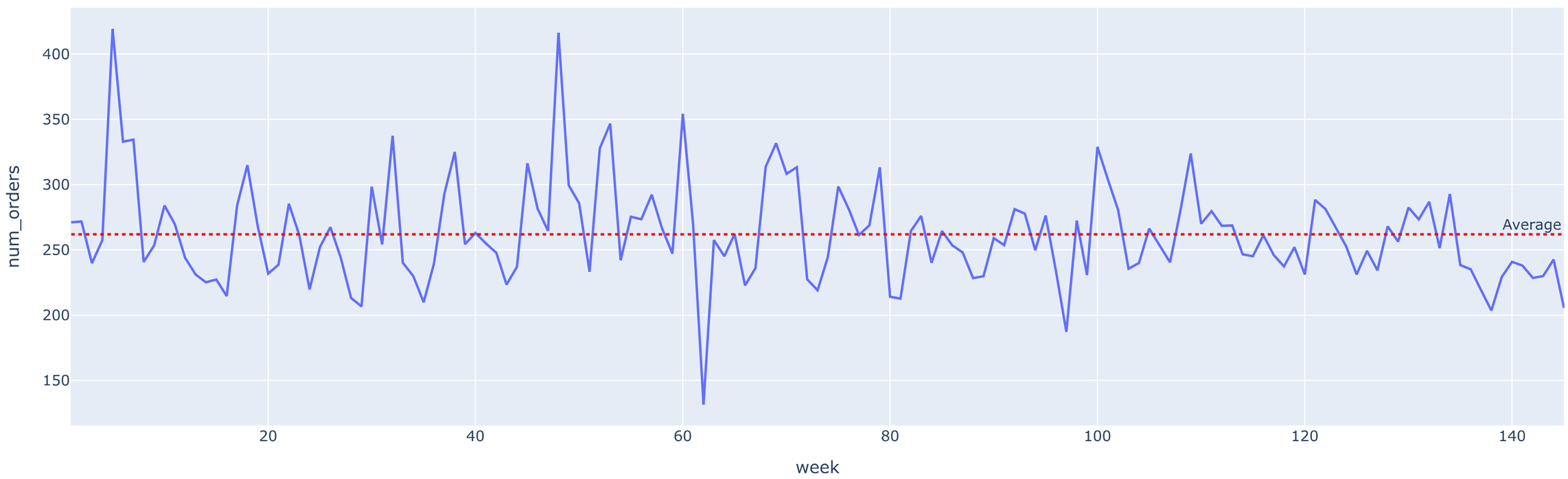
Finding which meal is contributed more to our business

```
px.pie(
    train.groupby('meal_id')['num_orders'].mean().reset_index(),
    names='meal_id',
    values='num_orders',
    hole=0.2,
    title='Average Number of Orders by Meal ID'
).update_layout(
    height=900,
    width=900
)
```



Finding average orders by week (The red line shows average of orders)

```
fig=px.line(
    train.groupby('week')['num_orders'].mean().reset_index(),
    x='week',
    y='num_orders',
    title='Average Number of Orders by Week'
)
fig.add_hline(train['num_orders'].mean(),line_dash="dot", line_color="red",annotation_text="Average")
fig.show()
```



Training and validating the Data

```
X = train.drop(['id', 'num_orders'], axis=1)
y = train['num_orders']


X_train, X_valid, y_train, y_valid = train_test_split(X, y, test_size=0.2, random_state=42)
```

Building the XGBoost Model and Fitting the training data to it

```
xgb_model = XGBRegressor(
    n_estimators=750,
    learning_rate=0.05,
    max_depth=10,
    subsample=0.8,
    colsample_bytree=0.8,
    random_state=42,
    n_jobs=-1
)


xgb_model.fit(X_train, y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=0.8, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             feature_weights=None, gamma=None, grow_policy=None,
             importance_type=None, interaction_constraints=None,
             learning_rate=0.05, max_bin=None, max_cat_threshold=None,
             max_cat_to_onehot=None, max_delta_step=None, max_depth=10,
             max_leaves=None, min_child_weight=None, missing=nan,
             monotone_constraints=None, multi_strategy=None, n_estimators=750,
             n_jobs=-1, num_parallel_tree=None, ...)
```

Validation of data and Checking Variation(Using R2 Score)

```
y_pred = xgb_model.predict(X_valid)
mse = mean_squared_error(y_valid, y_pred)
r2 = r2_score(y_valid, y_pred)

print("Validation MSE:", mse)
print("Validation R² Score:", r2)
print(f"Accuracy Percentage: {r2 * 100:.2f}%")
```

```
Validation MSE: 19059.189453125
Validation R² Score: 0.8750491142272949
Accuracy Percentage: 87.50%
```

Now we are predicting the orders for next 10 weeks by giving whole test dataset to model as input once

```
X_test = test.drop(['id'], axis=1)
predictions = xgb_model.predict(X_test)

output = pd.DataFrame({'id': test['id'], 'num_orders': predictions})
output.to_csv('xgb_output.csv', index=False)
```

Now, We are Performing the merge operation between Test Data and The data we predicted, so that we can analyze orders for next 10 weeks.

```
New_predicted_data=pd.read_csv("/content/xgb_output.csv")
Final_data=test.merge(New_predicted_data,on="id",how="inner")
Final_data['num_orders']=round(Final_data['num_orders'])
Final_data
```

	id	week	center_id	meal_id	checkout_price	base_price	emailer_for_promotion	homepage_featured	num_orders
0	1028232	146	55	1885	158.11	159.11	0	0	415.0
1	1127204	146	55	1993	160.11	159.11	0	0	48.0
2	1212707	146	55	2539	157.14	159.14	0	0	138.0
3	1082698	146	55	2631	162.02	162.02	0	0	13.0
4	1400926	146	55	1248	163.93	163.93	0	0	-15.0
...	...	...	...	...	...	...	...	...	...
32568	1250239	155	61	1543	482.09	484.09	0	0	70.0
32569	1039516	155	61	2304	483.09	483.09	0	0	43.0
32570	1158107	155	61	2664	322.07	323.07	0	0	283.0
32571	1444235	155	61	2569	322.07	323.07	0	0	346.0
32572	1291286	155	61	2490	276.45	276.45	0	0	148.0

32573 rows × 9 columns

Now Simply plotting the data for next 10 weeks

```
fig_pred=px.line(
    Final_data.groupby('week')['num_orders'].mean().reset_index(),
    x='week',
    y='num_orders',
    title='Number of Orders by Week (For next 10 weeks)'
)
fig_pred.add_hline(Final_data['num_orders'].mean(),line_dash="dot", line_color="red",annotation_text="Average")
fig_pred.show()
```