We read in the data

In [228]:
```
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = 20, 10
import pandas as pd
import numpy as np


day_hour_count = pd.read_csv("bikeshare_hour_count.csv")
day_hour_count
```

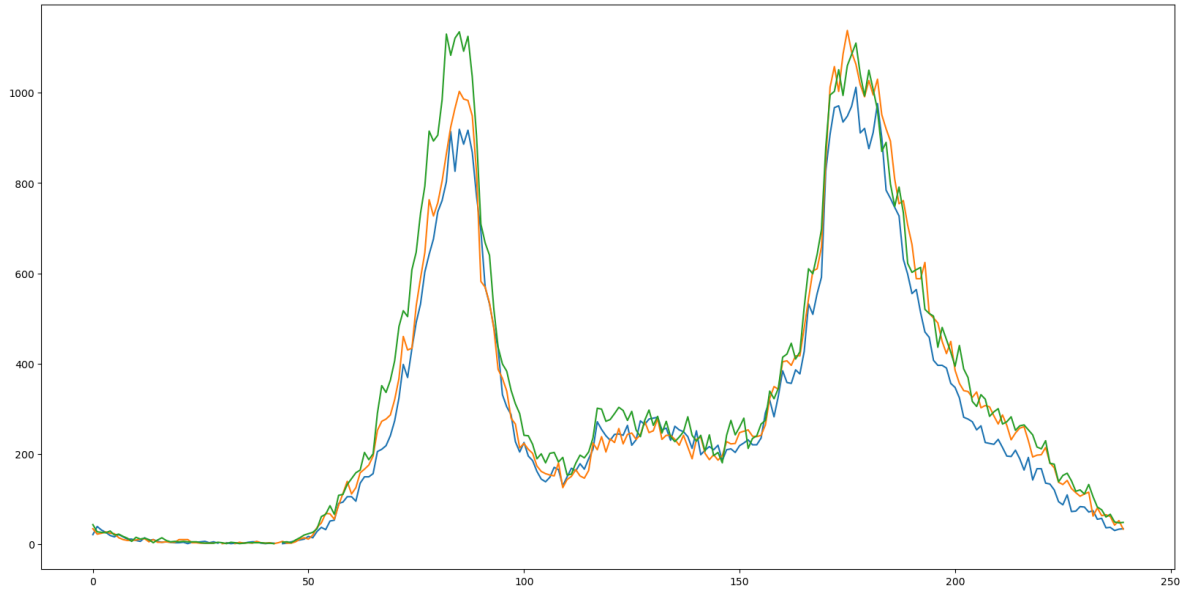
Out[228]:

	hour	monday	tuesday	wednesday	thursday	friday	saturday	sunday
0	0.0	21.0	34.0	43.0	47.0	51.0	89.0	106.0
1	0.1	39.0	22.0	27.0	37.0	56.0	87.0	100.0
2	0.2	31.0	24.0	26.0	42.0	50.0	98.0	77.0
3	0.3	26.0	27.0	25.0	29.0	52.0	99.0	87.0
4	0.4	19.0	24.0	29.0	29.0	50.0	98.0	69.0
...	...	...	...	...	...	...	...	...
235	23.5	36.0	65.0	60.0	94.0	80.0	93.0	28.0
236	23.6	37.0	61.0	66.0	100.0	81.0	95.0	28.0
237	23.7	30.0	42.0	49.0	80.0	101.0	105.0	27.0
238	23.8	33.0	52.0	47.0	79.0	91.0	93.0	24.0
239	23.9	34.0	33.0	48.0	65.0	105.0	111.0	23.0

240 rows × 8 columns

In [229]:
```
plt.figure(figsize=(20,10))
plt.plot(day_hour_count.index, day_hour_count["monday"])
plt.plot(day_hour_count.index, day_hour_count["tuesday"])
plt.plot(day_hour_count.index, day_hour_count["wednesday"])
```

Out[229]:
```
[<matplotlib.lines.Line2D at 0x25060d91070>]
```



Assignment 4

Explain the results in a **paragraph + charts** of to describe which model you'd recommend. This means show the data and the model's line on the same chart. The paragraph is a simple justification and comparison of the several models you tried.

1. Using the `day_hour_count` dataframe create 4 dataframes `monday`, `tuesday`, `saturday` and `sunday` that represent the data for those days. (hint: Monday is day=0)

In [230]:
```
monday = day_hour_count[["hour","monday"]].copy()
monday
```

Out[230]:

	hour	monday
0	0.0	21.0
1	0.1	39.0
2	0.2	31.0
3	0.3	26.0
4	0.4	19.0
...	...	...
235	23.5	36.0
236	23.6	37.0
237	23.7	30.0
238	23.8	33.0
239	23.9	34.0

240 rows × 2 columns

In [231]:
```
tuesday = day_hour_count[["hour", "tuesday"]].copy()
tuesday
```

Out[231]:

	hour	tuesday
0	0.0	34.0
1	0.1	22.0
2	0.2	24.0
3	0.3	27.0
4	0.4	24.0
...	...	...
235	23.5	65.0
236	23.6	61.0
237	23.7	42.0
238	23.8	52.0
239	23.9	33.0

240 rows × 2 columns

In [232]:
```
saturday = day_hour_count[["hour", "saturday"]].copy()
saturday
```

Out[232]:

	hour	saturday
0	0.0	89.0
1	0.1	87.0
2	0.2	98.0
3	0.3	99.0
4	0.4	98.0
...	...	...
235	23.5	93.0
236	23.6	95.0
237	23.7	105.0
238	23.8	93.0
239	23.9	111.0

240 rows × 2 columns

In [233]:
```
sunday = day_hour_count[["hour", "sunday"]].copy()
sunday
```

Out[233]:

	hour	sunday
0	0.0	106.0
1	0.1	100.0
2	0.2	77.0
3	0.3	87.0
4	0.4	69.0
...	...	...
235	23.5	28.0
236	23.6	28.0
237	23.7	27.0
238	23.8	24.0
239	23.9	23.0

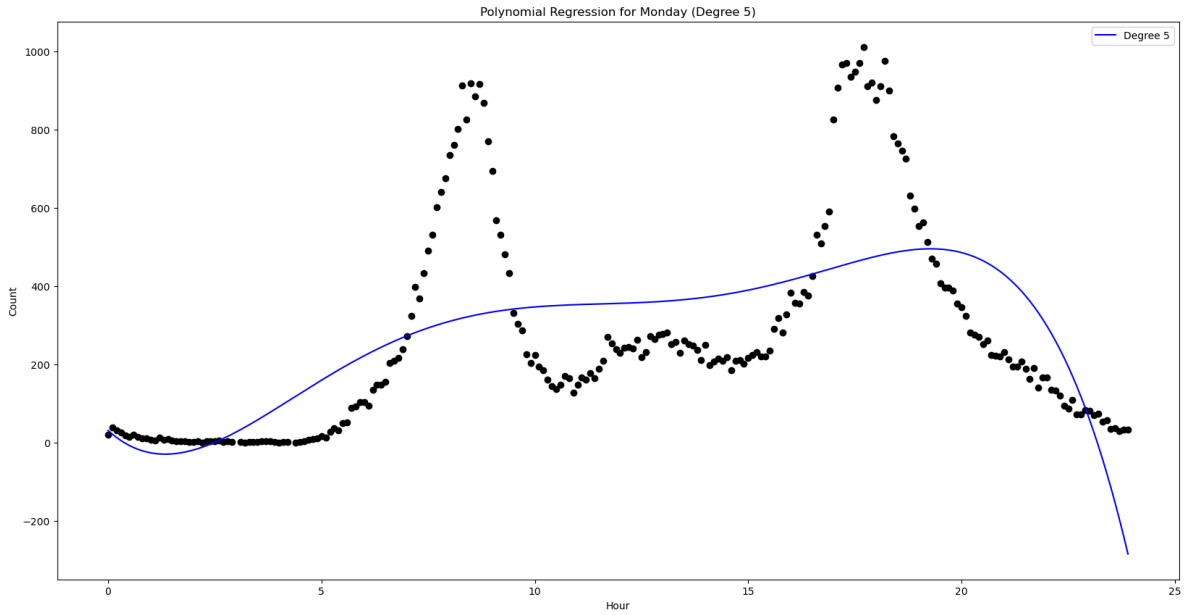
240 rows × 2 columns

2a. Create 3 models fit to (x=hour , y=monday) with varying polynomial degrees (choose from n=5,15,20).

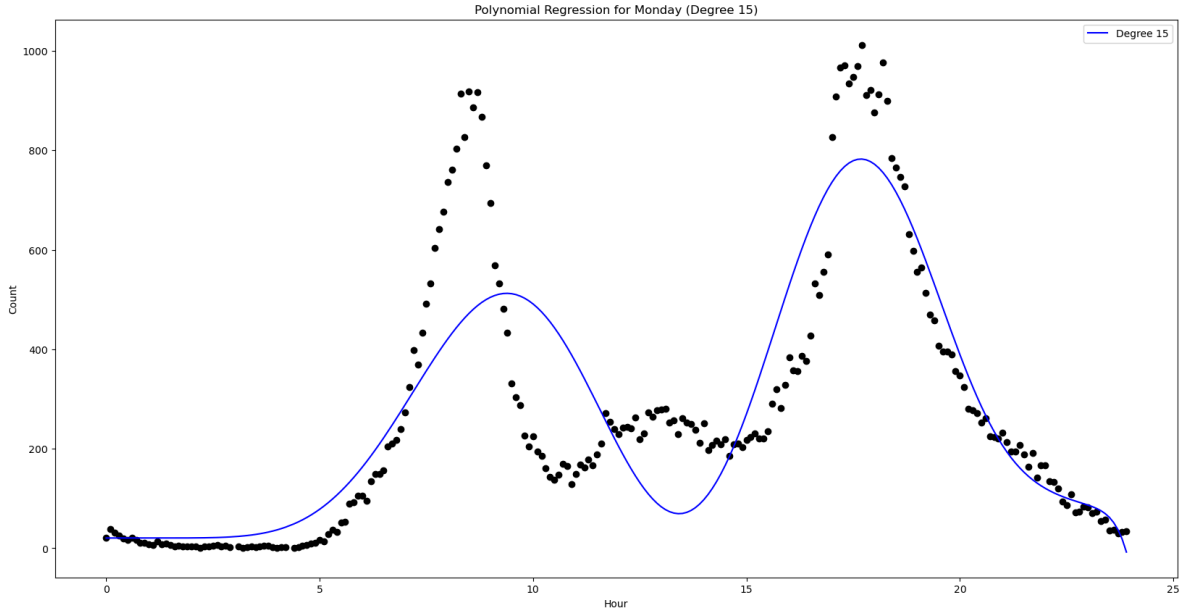
(Repeat for saturday below)

In [234]:
```
# Loading packages
import numpy as np
import matplotlib.pylab as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = 20, 10
from sklearn import linear_model, metrics
from sklearn.preprocessing import PolynomialFeatures
```

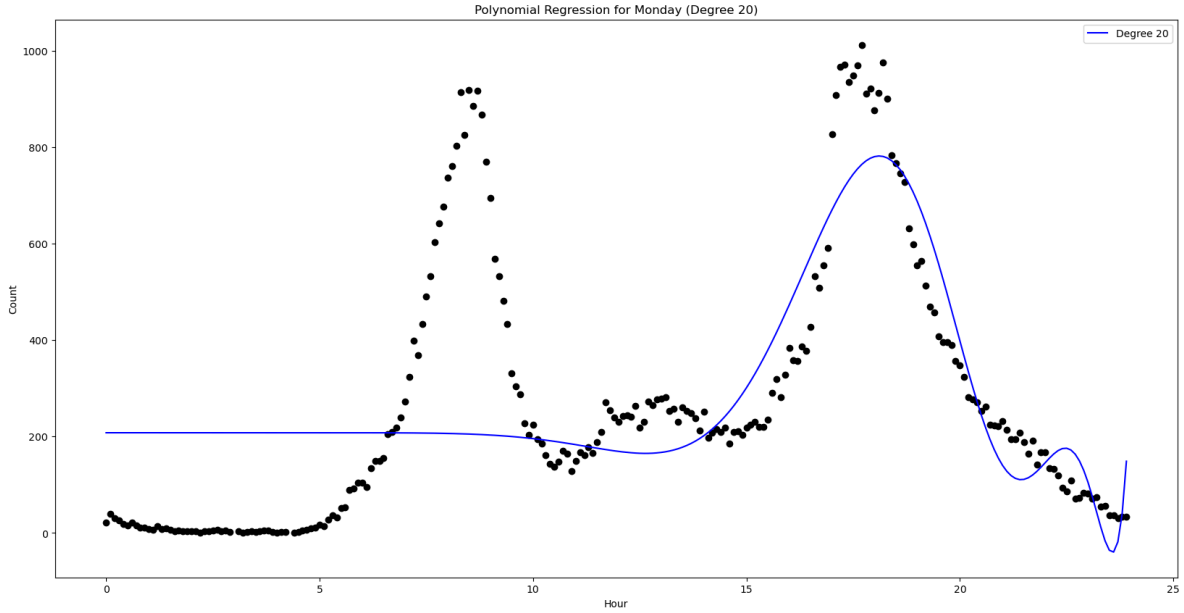
In [235]:
```
# Monday
monday_clean = monday.dropna()

# Preparing data
x_monday = monday_clean[['hour']]
y_monday = monday_clean[['monday']]

# n = 5
poly5 = PolynomialFeatures(degree=5)
mon_poly5 = poly5.fit_transform(x_monday)

linear5 = linear_model.LinearRegression()
linear5.fit(mon_poly5, y_monday)
y_poly5_pred = linear5.predict(mon_poly5)

# Plotting model
plt.figure(figsize=(20, 10))
plt.scatter(x_monday, y_monday, color='black')
plt.plot(x_monday, y_poly5_pred, label='Degree 5', color='blue')
plt.title('Polynomial Regression for Monday (Degree 5)')
plt.xlabel('Hour')
plt.ylabel('Count')
plt.legend()
plt.show()
```



In [236]:
```
# n = 15
poly15 = PolynomialFeatures(degree=15)
mon_poly15 = poly15.fit_transform(x_monday)

linear15 = linear_model.LinearRegression()
linear15.fit(mon_poly15, y_monday)
y_poly15_pred = linear15.predict(mon_poly15)

# Plotting model
plt.figure(figsize=(20, 10))
plt.scatter(x_monday, y_monday, color='black')
plt.plot(x_monday, y_poly15_pred, label='Degree 15', color='blue')
plt.title('Polynomial Regression for Monday (Degree 15)')
plt.xlabel('Hour')
plt.ylabel('Count')
plt.legend()
plt.show()
```



In [237]:
```
# n = 20
poly20 = PolynomialFeatures(degree=20)
mon_poly20 = poly20.fit_transform(x_monday)

linear20 = linear_model.LinearRegression()
linear20.fit(mon_poly20, y_monday)
y_poly20_pred = linear20.predict(mon_poly20)

# Plotting model
plt.figure(figsize=(20, 10))
plt.scatter(x_monday, y_monday, color='black')
plt.plot(x_monday, y_poly20_pred, label='Degree 20', color='blue')
plt.title('Polynomial Regression for Monday (Degree 20)')
plt.xlabel('Hour')
plt.ylabel('Count')
plt.legend()
plt.show()
```



In [238]:
```
# Comparing Mean Squared Errors for Monday
mon_MSE = (
    metrics.mean_squared_error(y_monday[-4:], linear5.predict(mon_poly5[-4:]))
    metrics.mean_squared_error(y_monday[-4:], linear15.predict(mon_poly15[-4:]
    metrics.mean_squared_error(y_monday[-4:], linear20.predict(mon_poly20[-4:]
)

# Comparing Mean Absolute Errors for Monday
mon_MAE = (
    metrics.mean_absolute_error(y_monday[-4:], linear5.predict(mon_poly5[-4:])
    metrics.mean_absolute_error(y_monday[-4:], linear15.predict(mon_poly15[-4:
    metrics.mean_absolute_error(y_monday[-4:], linear20.predict(mon_poly20[-4:
)

# Comparing Mean Absolute Percentage Errors for Monday
mon_MAPE = (
    metrics.mean_absolute_percentage_error(y_monday[-4:], linear5.predict(mon_
    metrics.mean_absolute_percentage_error(y_monday[-4:], linear15.predict(mon
    metrics.mean_absolute_percentage_error(y_monday[-4:], linear20.predict(mon
)
```

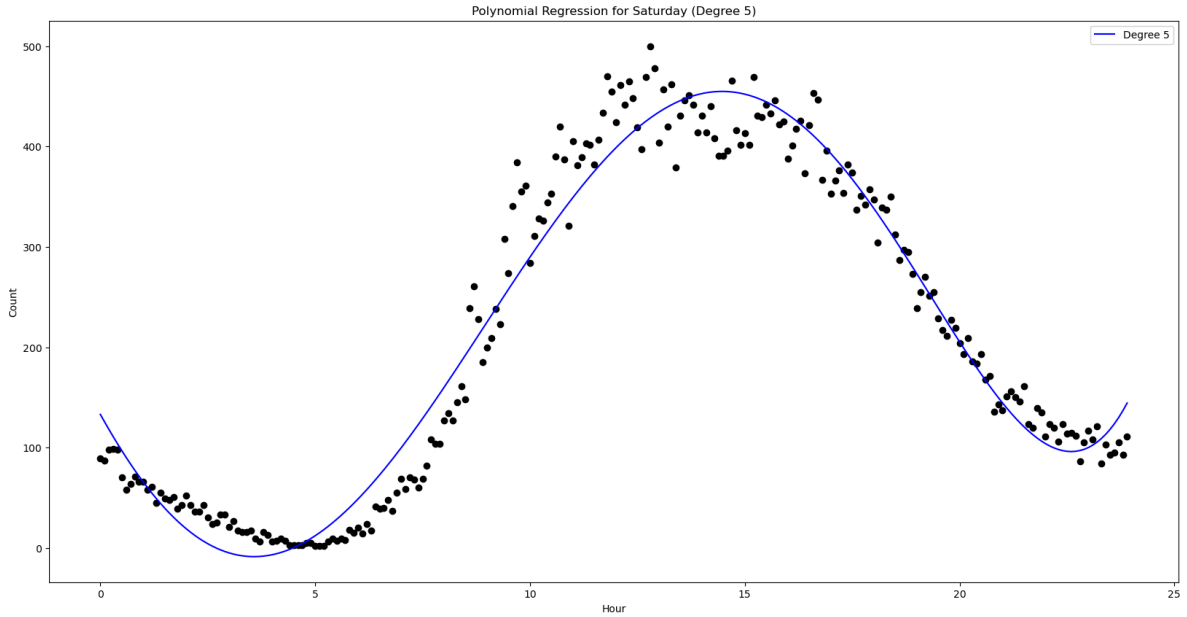
In [239]:
```
("MSE", mon_MSE,
"MAE", mon_MAE,
"MAPE", mon_MAPE)
```

Out[239]:
```
('MSE',
 (65685.37857653813, 557.2165130414387, 5350.196323439163),
 'MAE',
 (251.7760978188714, 19.66744112968445, 61.531613916158676),
 'MAPE',
 (7.573057477913766, 0.5803776060332144, 1.8129373118089216))
```

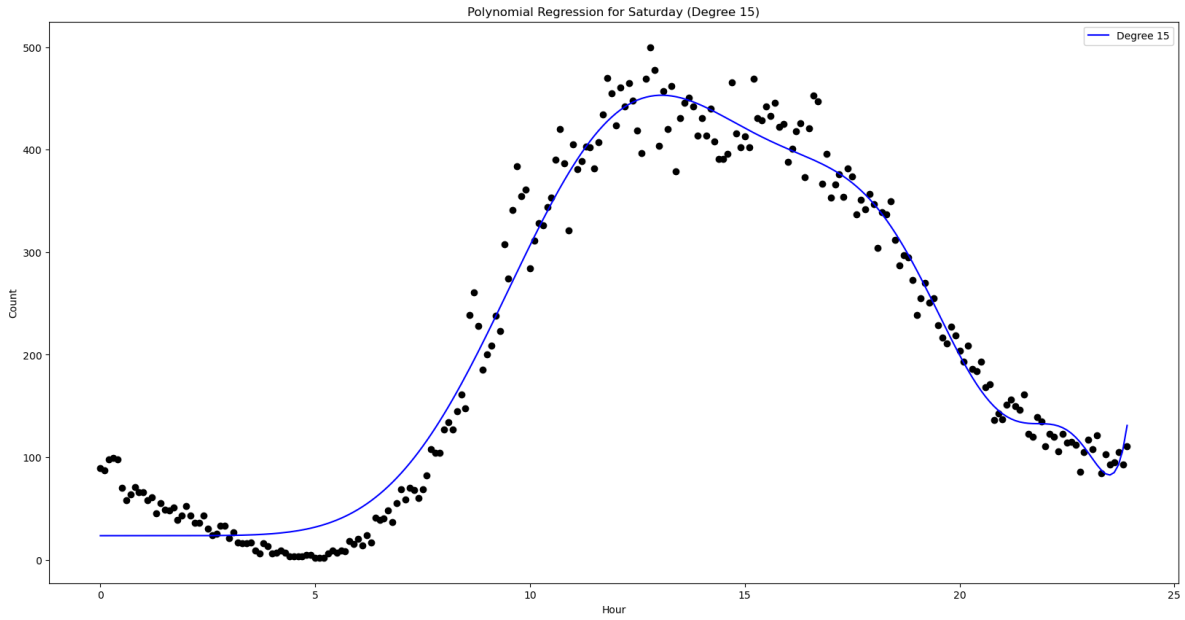
Interpretation: Based on the Mean Squared Errors, Mean Absolute Errors, and Mean Absolute Percentage Errors computed above for the Monday prediction, the polynomial with the degree of 15 seems to have the best fit. This model produced the lowest errors across all measurements by a significant margin. When analyzing the graphs, we also observe that the second model (degree of 15) closely aligns with the shape of the actual values. Although this model demonstrates better predictive quality later in the day, as evidened by closer alignment of the predicted and actual values, the model with a degree of 15 is the only one that accurately presents two peaks in activity throughout the day. Taking this into consideration, model 2 (degree = 15) is the best fit among the three models examined here.

2b. Repeat 2a for saturday

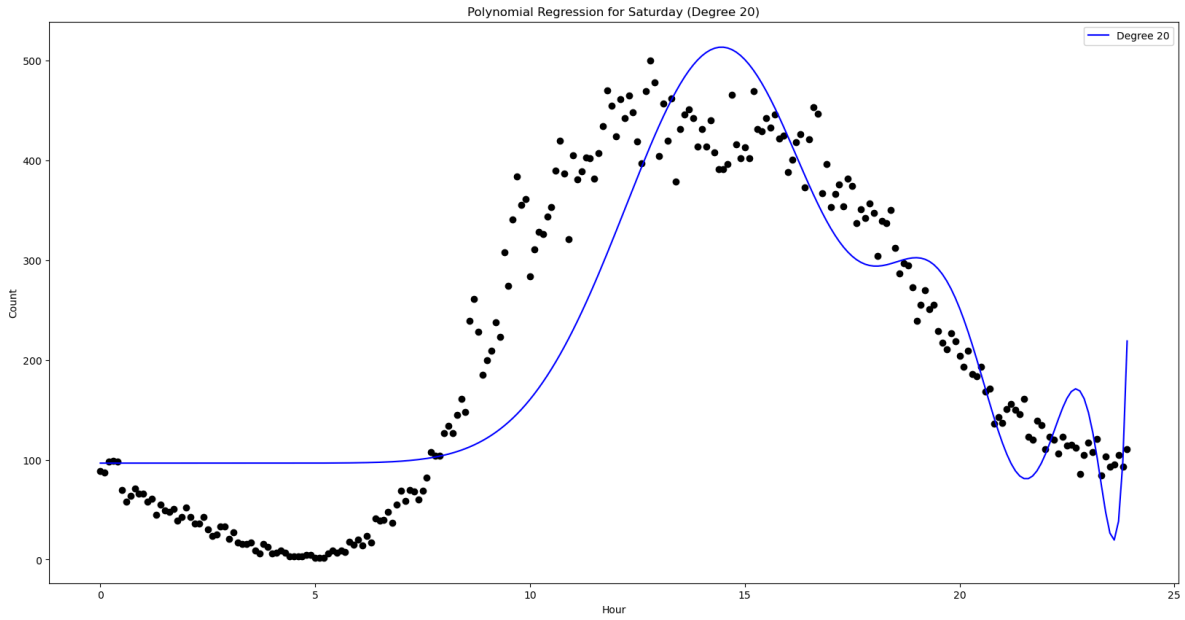
In [240]:
```
# Saturday
sat_clean = saturday.dropna()

# Preparing data
x_sat = sat_clean[['hour']]
y_sat = sat_clean[['saturday']]

# n = 5
poly5 = PolynomialFeatures(degree=5)
sat_poly5 = poly5.fit_transform(x_sat)

linear5 = linear_model.LinearRegression()
linear5.fit(sat_poly5, y_sat)
y_poly5_pred = linear5.predict(sat_poly5)

# Plotting model
plt.figure(figsize=(20, 10))
plt.scatter(x_sat, y_sat, color='black')
plt.plot(x_sat, y_poly5_pred, label='Degree 5', color='blue')
plt.title('Polynomial Regression for Saturday (Degree 5)')
plt.xlabel('Hour')
plt.ylabel('Count')
plt.legend()
plt.show()
```



In [241]:
```
# n = 15
poly15 = PolynomialFeatures(degree=15)
sat_poly15 = poly15.fit_transform(x_sat)

linear15 = linear_model.LinearRegression()
linear15.fit(sat_poly15, y_sat)
y_poly15_pred = linear15.predict(sat_poly15)

# Plotting model
plt.figure(figsize=(20, 10))
plt.scatter(x_sat, y_sat, color='black')
plt.plot(x_sat, y_poly15_pred, label='Degree 15', color='blue')
plt.title('Polynomial Regression for Saturday (Degree 15)')
plt.xlabel('Hour')
plt.ylabel('Count')
plt.legend()
plt.show()
```



In [242]:
```
# n = 20
poly20 = PolynomialFeatures(degree=20)
sat_poly20 = poly20.fit_transform(x_sat)

linear20 = linear_model.LinearRegression()
linear20.fit(sat_poly20, y_sat)
y_poly20_pred = linear20.predict(sat_poly20)

# Plotting model
plt.figure(figsize=(20, 10))
plt.scatter(x_sat, y_sat, color='black')
plt.plot(x_sat, y_poly20_pred, label='Degree 20', color='blue')
plt.title('Polynomial Regression for Saturday (Degree 20)')
plt.xlabel('Hour')
plt.ylabel('Count')
plt.legend()
plt.show()
```



In [243]:
```
# Comparing Mean Squared Errors for Saturday
sat_MSE = (
    metrics.mean_squared_error(y_sat[-4:], linear5.predict(sat_poly5[-4:])),
    metrics.mean_squared_error(y_sat[-4:], linear15.predict(sat_poly15[-4:])),
    metrics.mean_squared_error(y_sat[-4:], linear20.predict(sat_poly20[-4:]))
)

# Comparing Mean Absolute Errors for Saturday
sat_MAE = (
    metrics.mean_absolute_error(y_sat[-4:], linear5.predict(sat_poly5[-4:])),
    metrics.mean_absolute_error(y_sat[-4:], linear15.predict(sat_poly15[-4:]))
    metrics.mean_absolute_error(y_sat[-4:], linear20.predict(sat_poly20[-4:]))
)

# Comparing Mean Absolute Percentage Errors for Saturday
sat_MAPE = (
    metrics.mean_absolute_percentage_error(y_sat[-4:], linear5.predict(sat_pol
    metrics.mean_absolute_percentage_error(y_sat[-4:], linear15.predict(sat_po
    metrics.mean_absolute_percentage_error(y_sat[-4:], linear20.predict(sat_po
)
```

In [244]:
```
("MSE", sat_MSE,
"MAE", sat_MAE,
"MAPE", sat_MAPE)
```

Out[244]:
```
('MSE',
 (1109.1255253423553, 212.7431614999037, 5451.570223780617),
 'MAE',
 (32.537769210340514, 14.111050337553024, 63.74161195755005),
 'MAPE',
 (0.3261409311138943, 0.1386668537798772, 0.6135771451070261))
```

Interpretation: Similar to the Monday model, the polynomial model with the degree of 15 was also the best fitting model for Saturday, as evidenced by this model producing the smallest mean squared error, mean absolute error, and mean absolute percentage errors of all 3 models examined here. This is further illustrated in the plot of this model, which shows a closer fit of the predicted to actual values, particularly after 8 AM.

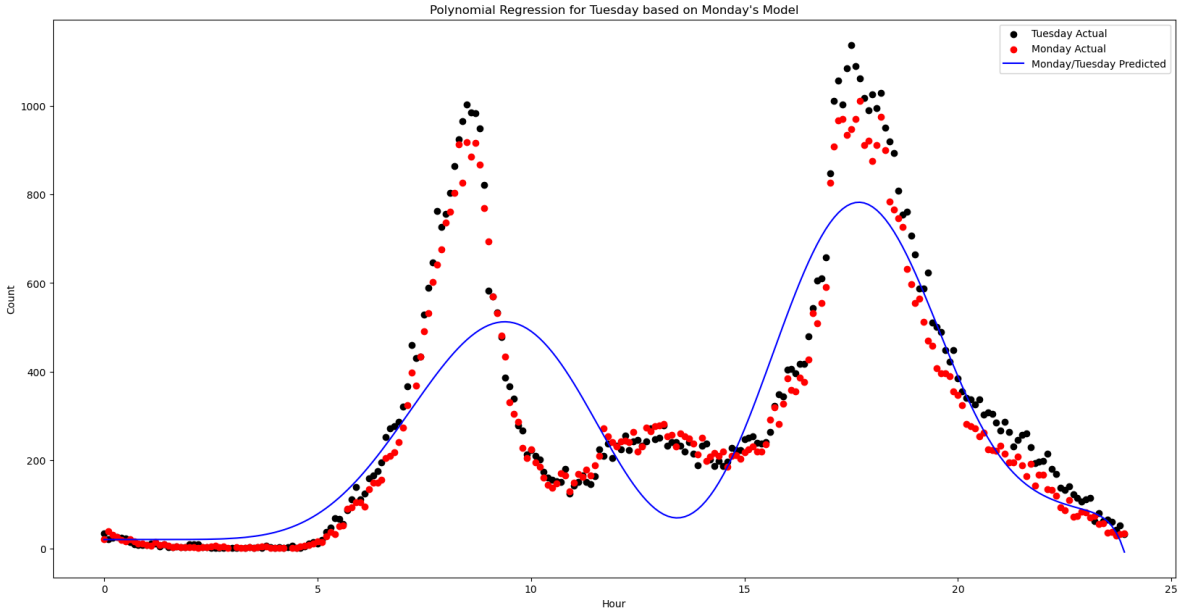
3. Using the best `monday` model's prediction, determine the errors (MSE, MAE, MAPE) between the prediction with the `monday` and `tuesday` datasets

Repeat for `saturday` / `sunday`

In [245]:
```
# Dropping missing values for Tuesday and Sunday datasets

# Tueday
tues_clean = tuesday.dropna()

# Sunday
sun_clean = sunday.dropna()
```

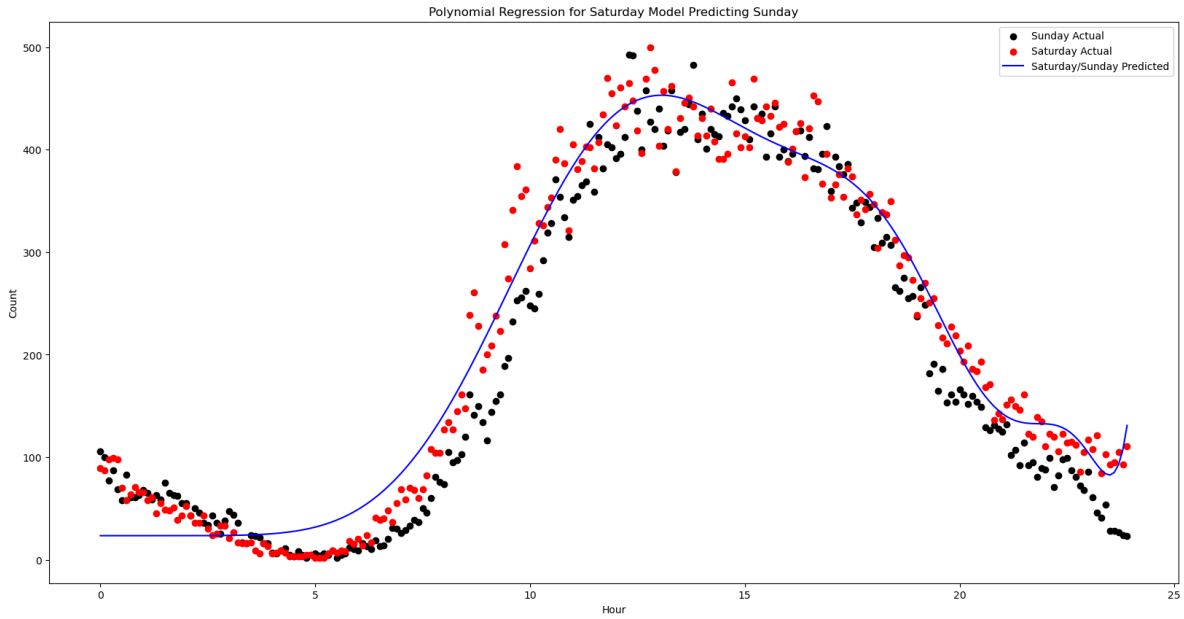
In [246]:
```
# Monday/Tuesday

# Predicting Tuesday based on the best Monday model
x_tues = tues_clean[['hour']]
y_tues = tues_clean[['tuesday']]

# Polynomial transformation
poly15 = PolynomialFeatures(degree=15)

# Monday model and prediction for Tuesday
x_monday_poly15 = poly15.fit_transform(x_monday)
x_tuesday_poly15 = poly15.transform(x_tues)

# Training Monday model
linear15_mon = linear_model.LinearRegression().fit(x_monday_poly15, y_monday)

# Predicting Tuesday
y_tuesday_pred = linear15_mon.predict(x_tuesday_poly15)

# Calculating errors for Monday/Tuesday
mse_mon_tues = metrics.mean_squared_error(y_tues, y_tuesday_pred)
mae_mon_tues = metrics.mean_absolute_error(y_tues, y_tuesday_pred)
mape_mon_tues = metrics.mean_absolute_percentage_error(y_tues, y_tuesday_pred)

print("Monday/Tuesday Errors:")
print("MSE:", mse_mon_tues)
print("MAE:", mae_mon_tues)
print("MAPE:", mape_mon_tues)

# Plotting Monday and Tuesday
plt.figure(figsize=(20, 10))
plt.scatter(x_tues, y_tues, color='black', label='Tuesday Actual')
plt.scatter(x_monday, y_monday, color = 'red', label = 'Monday Actual')
plt.plot(x_tues, y_tuesday_pred, label='Monday/Tuesday Predicted', color='blue
plt.title("Polynomial Regression for Tuesday based on Monday's Model")
plt.xlabel('Hour')
plt.ylabel('Count')
plt.legend()
plt.show()
```

```
Monday/Tuesday Errors:
MSE: 23866.3650452427
MAE: 105.97962678403267
MAPE: 1.9135031254242623
```


Polynomial Regression for Tuesday based on Monday's Model
Tuesday Actual
Monday Actual
Monday/Tuesday Predicted
Count
Hour
0
200
400
600
800
1000
0
5
10
15
20
25

In [247]:
```
# Saturday model and prediction for Sunday
x_saturday_poly15 = poly15.fit_transform(x_sat)
x_sunday_poly15 = poly15.transform(x_sun)

# Training Saturday model
linear15_sat = linear_model.LinearRegression().fit(x_saturday_poly15, y_sat)

# Predicting Sunday
y_sunday_pred = linear15_sat.predict(x_sunday_poly15)
y_sat_pred = linear15_sat.predict(x_saturday_poly15)

# Calculating errors for Saturday/Sunday
mse_sat_sun = metrics.mean_squared_error(y_sun, y_sunday_pred)
mae_sat_sun = metrics.mean_absolute_error(y_sun, y_sunday_pred)
mape_sat_sun = metrics.mean_absolute_percentage_error(y_sun, y_sunday_pred)

print("Saturday/Sunday Errors:")
print("MSE:", mse_sat_sun)
print("MAE:", mae_sat_sun)
print("MAPE:", mape_sat_sun)

# Plotting Saturday and Sunday
plt.figure(figsize=(20, 10))
plt.scatter(x_sun, y_sun, color='black', label='Sunday Actual')
plt.scatter(x_sat, y_sat, color = 'red', label = "Saturday Actual")
plt.plot(x_sun, y_sunday_pred, label='Saturday/Sunday Predicted', color='blue'
plt.xlabel('Hour')
plt.ylabel('Count')
plt.legend()
plt.title('Polynomial Regression for Saturday Model Predicting Sunday')
plt.show()
```

```
Saturday/Sunday Errors:
MSE: 1638.7496207600425
MAE: 34.648282964476145
MAPE: 0.9312482963364911
```



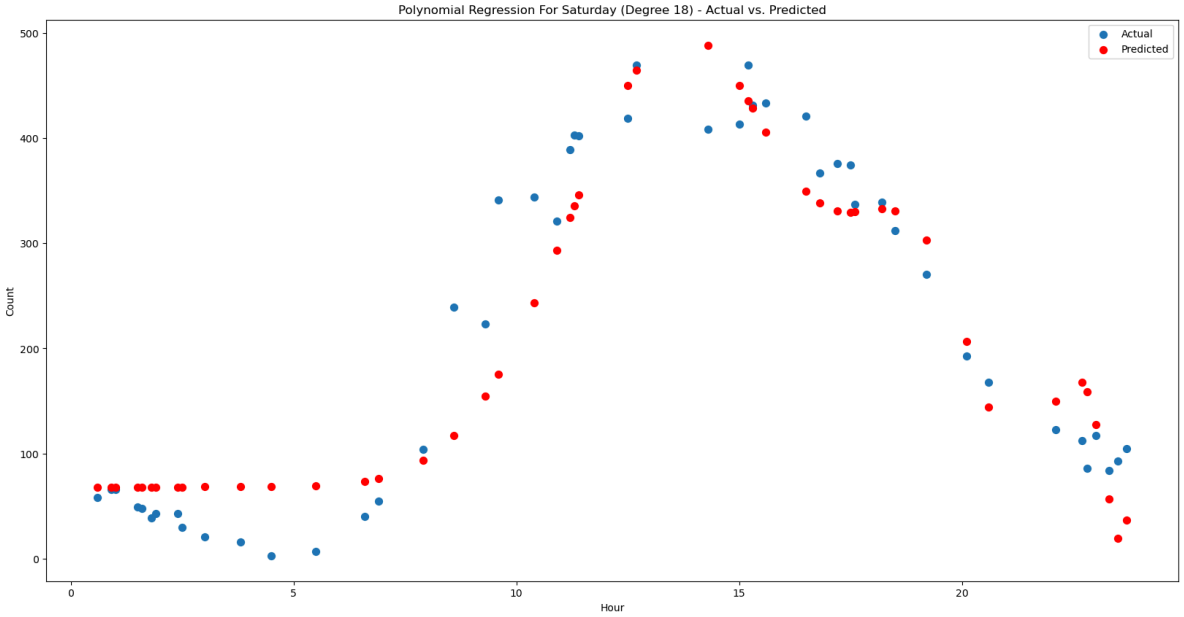
Both Monday and Saturday models applied reasonably well to the Tuesday and Sunday data respectively. However, for both models, all measurements of error slightly increased, suggesting a worse fit. Despite this decrease in performance, we see that Mondays and Tuesdays as well as Saturdays and Sundays seem to follow a similar pattern, suggesting similarities within weekdays and weekend days. With that said, the Monday/Tuesday model could benefit from further refinement - considering the curve of the regression line is somewhat displaced from the actual values. However, for Saturday and Sunday, the regression line is relatively accurate, performing slightly less well for earlier hours of the day (as previously noted).

4. With `saturday`, use `train_test_split` to create training and test sets and build a model. Create predictions using the xtest from and determine the errors between these predictions and the ytest (MSE, MAE, MAPE).

repeat for `monday`

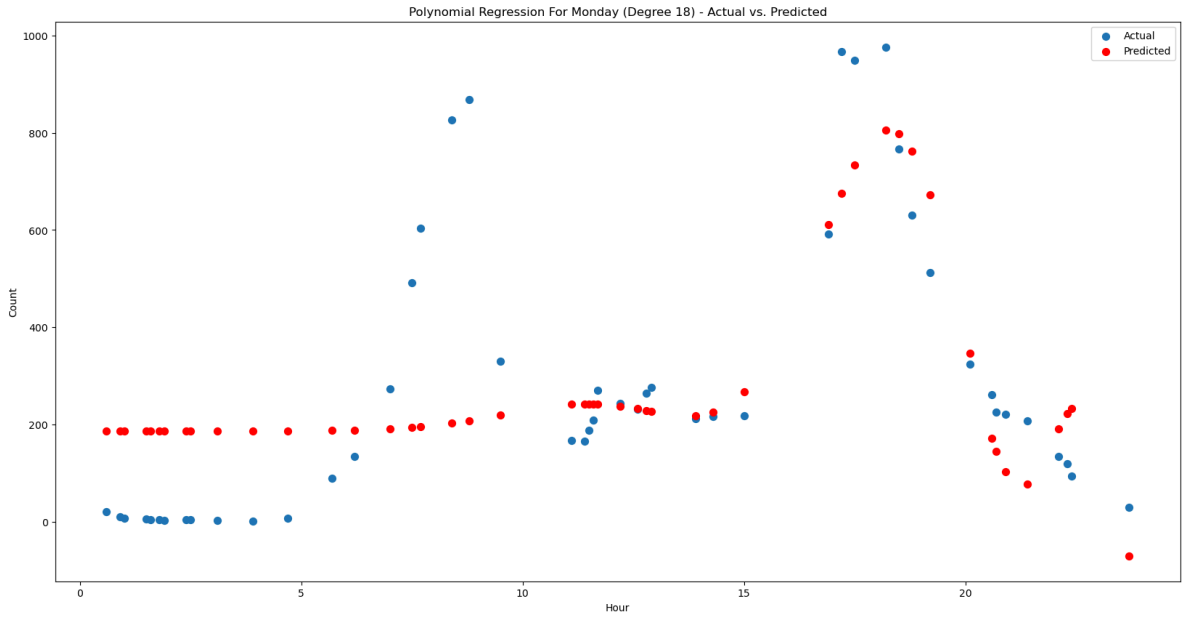
In [248]:
```
# Loading package
from sklearn.model_selection import train_test_split

# Splitting Saturday into training and testing sets
x_sat_train, x_sat_test, y_sat_train, y_sat_test = train_test_split(x_sat, y_s

# Modifying polynomial
x_sat_train18 = PolynomialFeatures(degree=18).fit_transform(x_sat_train)
x_sat_test18 = PolynomialFeatures(degree=18).fit_transform(x_sat_test)

# Creating model
linear18 = linear_model.LinearRegression().fit(x_sat_train18, y_sat_train)

# Plotting model
size = 50
plt.scatter(x_sat_test, y_sat_test, s=size, label = 'Actual')
plt.scatter(x_sat_test, linear18.predict(x_sat_test18), c='r', s=size, label =
plt.xlabel('Hour')
plt.ylabel('Count')
plt.legend()
plt.title('Polynomial Regression For Saturday (Degree 18) - Actual vs. Predict

# Determining errors
    # Mean Squared Error
("Mean Squared Error:",
 metrics.mean_squared_error(y_sat_test, linear18.predict(x_sat_test18)),
 "Mean Absolute Error",
metrics.mean_absolute_error(y_sat_test, linear18.predict(x_sat_test18)),
 "Mean Absolute Percentage Error",
metrics.mean_absolute_percentage_error(y_sat_test, linear18.predict(x_sat_test
)
```

Out[248]:
```
('Mean Squared Error:',
 2755.021189031784,
 'Mean Absolute Error',
 41.3327209469954,
 'Mean Absolute Percentage Error',
 1.0175549281563545)
```


Polynomial Regression For Saturday (Degree 18) - Actual vs. Predicted
Actual
Predicted
Count
Hour
0
100
200
300
400
500
0
5
10
15
20

In [249]:
```
# Splitting Monday into training and testing sets
x_mon_train, x_mon_test, y_mon_train, y_mon_test = train_test_split(x_monday,

# Modifying polynomial
x_mon_train18 = PolynomialFeatures(degree=18).fit_transform(x_mon_train)
x_mon_test18 = PolynomialFeatures(degree=18).fit_transform(x_mon_test)

# Creating model
linear18 = linear_model.LinearRegression().fit(x_mon_train18, y_mon_train)

# Plotting model
size = 50
plt.scatter(x_mon_test, y_mon_test, s=size, label = 'Actual')
plt.scatter(x_mon_test, linear18.predict(x_mon_test18), c='r', s=size, label =
plt.xlabel('Hour')
plt.ylabel('Count')
plt.legend()
plt.title('Polynomial Regression For Monday (Degree 18) - Actual vs. Predicted

# Determining errors
    # Mean Squared Error
("Mean Squared Error:",
 metrics.mean_squared_error(y_mon_test, linear18.predict(x_mon_test18)),
 "Mean Absolute Error",
metrics.mean_absolute_error(y_mon_test, linear18.predict(x_mon_test18)),
 "Mean Absolute Percentage Error",
metrics.mean_absolute_percentage_error(y_mon_test, linear18.predict(x_mon_test
)
```

Out[249]:
```
('Mean Squared Error:',
 37833.747704308924,
 'Mean Absolute Error',
 141.24055381850252,
 'Mean Absolute Percentage Error',
 10.588250566855011)
```



Summary

In this assignment, we trained three different polynomial regression models in attempt to predict bikeshare rentals by the hour on certain days of the week. Specifically, we trained polynomial regression models with degrees of 5, 15, and 20 using data from Monday and Saturday. For both Monday and Saturday, the degree 15 model was the best fit, as evidenced by the lowest error terms (i.e., mean-squared error, mean absolute error, and mean absolute percentage error) across all 3 models. Taking this into consideration, the degree 15 model that was trained on the Monday and Saturday data was then applied to predict Tuesday and Sunday rentals, respectively. Although these models fit reasonably well, error did increase - suggesting slight differences by days of the week. However, it is important to note that Monday and Tuesday were far more similar to one another than to Saturday or Sunday. Finally, Monday and Saturday test dataests were split into training and testing datasets in part 4. Using a polynomial degree of 18, the errors of these models drastically increased from those seen in a degree of 15. Ultimately, this suggests that of the models evaluated here, a polynomial degree of 15 is the best fit.

In [ ]: