In [50]:
```
#Assignment 1: Data Collection, Visualization, and Pre-processing
#Name: Uday Bhaskar Valapadasu
#ID: 11696364

#Downloaded the diabetes.csv
```

In [51]:
```
#Import Statements

import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
# Suppress the FutureWarning
warnings.simplefilter(action='ignore', category=FutureWarning)
```

In [55]:
```
#Loading the dataset
patients_list = pd.read_csv("diabetes.csv")
patients_list
```

Out[55]:

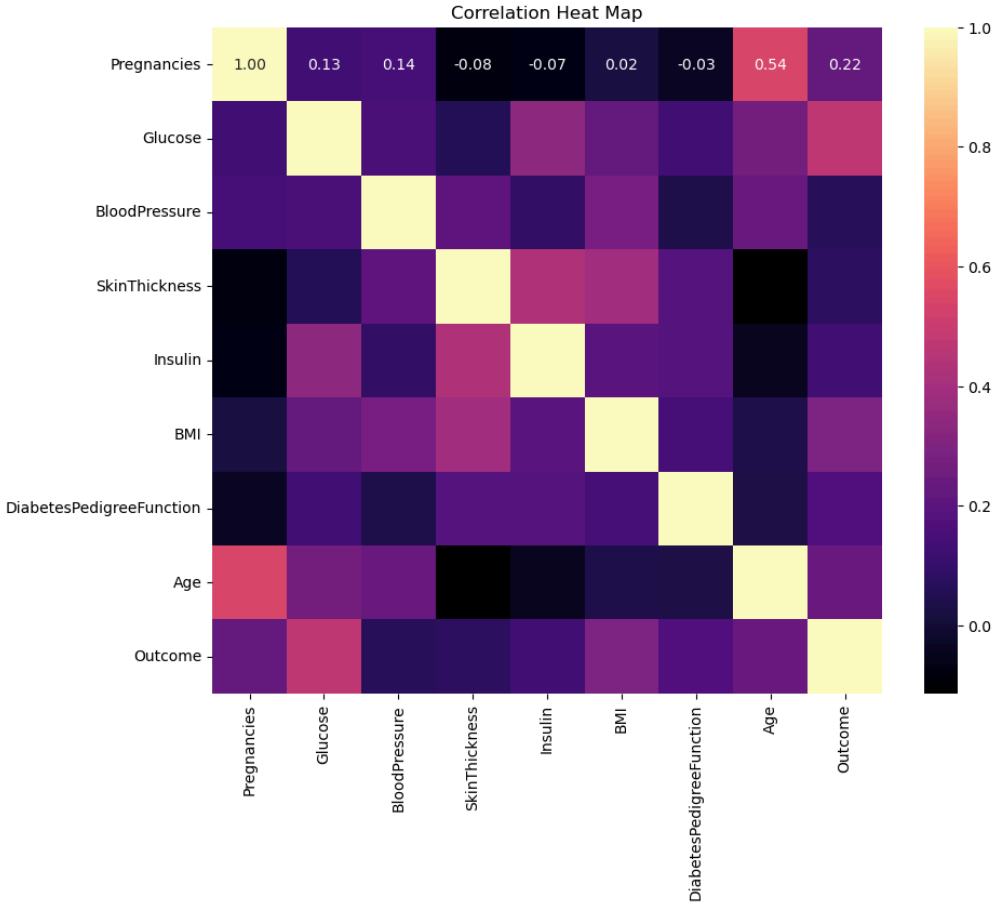
	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
...	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

768 rows × 9 columns

In [56]:
```
# Filtered the records having BMI interval [27, 37] and Outcome is 1.
# Also I am considering the interval values 27 & 37 as included as it is not mentioned.
def modified_patient_records(records):
   return records[(records['Outcome'] == 1) & ((records['BMI'] >=27) & (records['BMI'] <=37))]

# Print the filtered records
print(modified_patient_records(patients_list))
```

```
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35        0  33.6
6              3       78             50             32       88  31.0
8              2      197             70             45      543  30.5
13             1      189             60             23      846  30.1
15             7      100              0              0        0  30.0
..           ...      ...            ...            ...      ...   ...
754            8      154             78             32        0  32.4
755            1      128             88             39      110  36.5
757            0      123             72              0        0  36.3
759            6      190             92              0        0  35.5
766            1      126             60              0        0  30.1

     DiabetesPedigreeFunction  Age  Outcome
0                       0.627   50        1
6                       0.248   26        1
8                       0.158   53        1
13                      0.398   59        1
15                      0.484   32        1
..                        ...  ...      ...
754                     0.443   45        1
755                     1.057   37        1
757                     0.258   52        1
759                     0.278   66        1
766                     0.349   47        1

[161 rows x 9 columns]
```

In [57]:
```
# High importance is defined as the value being close to 1.0 but not equal to 1.0. Sometimes 0.2 is of high importance, and other times
# 0.02 is important.

# Creating a correlation matrix quantifies the relationships between variables, providing values that indicate the strength and direction
# of these relationships. A heatmap then visually represents this matrix, making it easier to identify and analyze patterns and
# important features in the data.

# Create a heat map using seaborn library
corr = patients_list.corr()
plt.figure(figsize=(10, 8))
heat_map = sns.heatmap(corr, annot=True, fmt=".2f", cmap='magma')
plt.title('Correlation Heat Map')
plt.show()
```



In [58]:
```
# After analyzing the correlation heatmap, I observed that the features
#1.'Age' and 'Pregnancies'
#2.'Pregnancies' and 'Age'
# exhibit a relatively high correlation value of approximately 0.54, which is considered close to 1.0 and indicates a strong positive
# correlation between these features and the target variable (e.g., diabetes outcome).
```

In [59]:
```
#1. Create diabetes_df (a dataframe for entire data)
diabetes_df = pd.read_csv("diabetes.csv")
```

In [60]:
```
#2. Drop records with missing data (0, empty column values, NaN, etc.).
# For each command, print "# of records removed".
# Print "zero records removed" if the condition is not met.

initial_dataset_count = len(diabetes_df)

# Drop rows with NaN values
diabetes_df = diabetes_df.dropna()
after_dropped_nan_count = len(diabetes_df)
print(f"# of records removed (NaN): {initial_dataset_count - after_dropped_nan_count}")

# Drop rows with empty strings
diabetes_df = diabetes_df[~(diabetes_df == '').any(axis=1)]
after_dropped_empty_string_count = len(diabetes_df)
print(f"# of records removed (empty values): {after_dropped_nan_count - after_dropped_empty_string_count}")

total_removed = initial_dataset_count - after_dropped_empty_string_count
if total_removed == 0:
    print("zero records removed")
else:
    print(f"Total # of records removed: {total_removed}")
```

```
# of records removed (NaN): 0
# of records removed (empty values): 0
zero records removed
```

In [61]:
```
# Here I have performed the replace insulin before drop rows with 0 value operation, Because according to the question if, we do the drop rows with
# missing values in step 2 i.e 0 then there is no use of doing the 3rd step i.e 'Replace column values where Insulin is 0 with 150'
# because the rows with 0 values are already deleted in step 2. We are trying to replace 0 with 150 in column "Insulin".


# Replace Insulin values that are 0 with 150
insulinRCount = (diabetes_df['Insulin'] == 0).sum()
diabetes_df.loc[diabetes_df['Insulin'] == 0, 'Insulin'] = 150
print(f"# of records were updated with new data for insulin: {insulinRCount}")

initial_count = len(diabetes_df)

# Drop rows with 0 values
diabetes_df = diabetes_df[~(diabetes_df == 0).any(axis=1)]
after_dropping_zero_count = len(diabetes_df)
print(f"# of records removed (0 values): {initial_count - after_dropping_zero_count}")
```

```
# of records were updated with new data for insulin: 374
# of records removed (0 values): 618
```

In [49]:
```
#4.Rename column 'Outcome' to 'Target'
diabetes_df = diabetes_df.rename(columns={'Outcome': 'Target'})

# Writing the updated dataframe and Saving it to the updated DataFrame to a new CSV file
diabetes_df.to_csv('diabetes_df.csv', index=False)
```

```
# Print the updated DataFrame
print("\nUpdated DataFrame:")
print(diabetes_df)
```

```
Updated DataFrame:
     Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0              6      148             72             35      150  33.6
6              3       78             50             32       88  31.0
8              2      197             70             45      543  30.5
13             1      189             60             23      846  30.1
14             5      166             72             19      175  25.8
..           ...      ...            ...            ...      ...   ...
746            1      147             94             41      150  49.3
748            3      187             70             22      200  36.4
754            8      154             78             32      150  32.4
755            1      128             88             39      110  36.5
761            9      170             74             31      150  44.0

     DiabetesPedigreeFunction  Age  Target
0                       0.627   50       1
6                       0.248   26       1
8                       0.158   53       1
13                      0.398   59       1
14                      0.587   51       1
..                        ...  ...     ...
746                     0.358   27       1
748                     0.408   36       1
754                     0.443   45       1
755                     1.057   37       1
761                     0.403   43       1

[150 rows x 9 columns]
```