In [1]:
```
import os
import pandas as pd
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import pandas as pd
import matplotlib.pyplot as plt
from PIL import Image
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
```

Loading dataset and assigning training and test datasets and lables(cancer vs non cancer)

In [2]:
```
os.chdir('/Users/evelynhaskins/Downloads/histopathologic-cancer-detection')
```

In [3]:
```
base_path = "/Users/evelynhaskins/Downloads/histopathologic-cancer-detectior
train_dir = os.path.join(base_path, "train")
test_dir = os.path.join(base_path, "test")

train_labels_path = os.path.join(base_path, "train_labels.csv")
labels = pd.read_csv(train_labels_path)
```

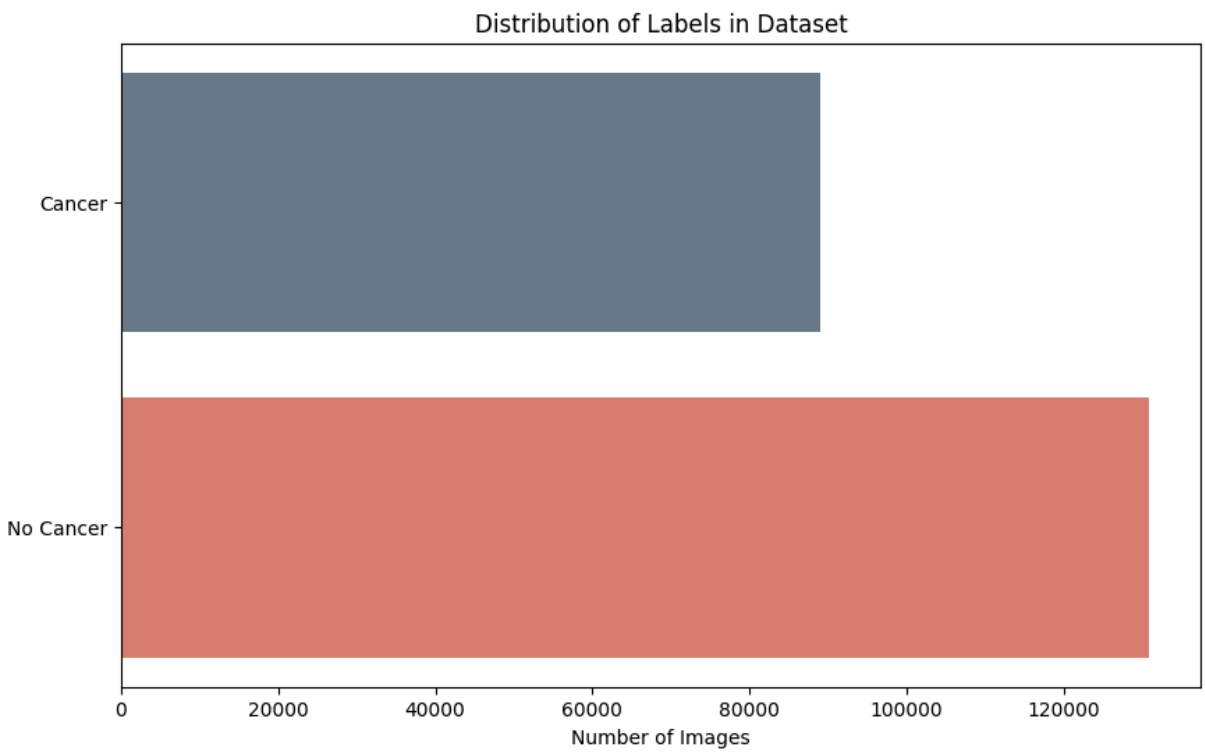
Looking into the dataset and seeing what the labels look like (0 and 1 for cancer vs non cancer)

In [4]:
```
print("Training Data Overview:")
print(labels.info())
print(labels.head())

# Checking the sizes of the train and test directories
print("\nTrain Size: {}".format(len(os.listdir(train_dir))))
print("Test Size: {}".format(len(os.listdir(test_dir))))

# Display the distribution of labels
labels_count = labels.label.value_counts()
```

```
Training Data Overview:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 220025 entries, 0 to 220024
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   id      220025 non-null  object
 1   label   220025 non-null  int64
dtypes: int64(1), object(1)
memory usage: 3.4+ MB
None
                                         id  label
0  f38a6374c348f90b587e046aac6079959adf3835      0
1  c18f2d887b7ae4f6742ee445113fa1aef383ed77      1
2  755db6279dae599ebb4d39a9123cce439965282d      0
3  bc3f0c64fb968ff4a8bd33af6971ecae77c75e08      0
4  068aba587a4950175d04c680d38943fd488d6a9d      0

Train Size: 220025
Test Size: 57458
```

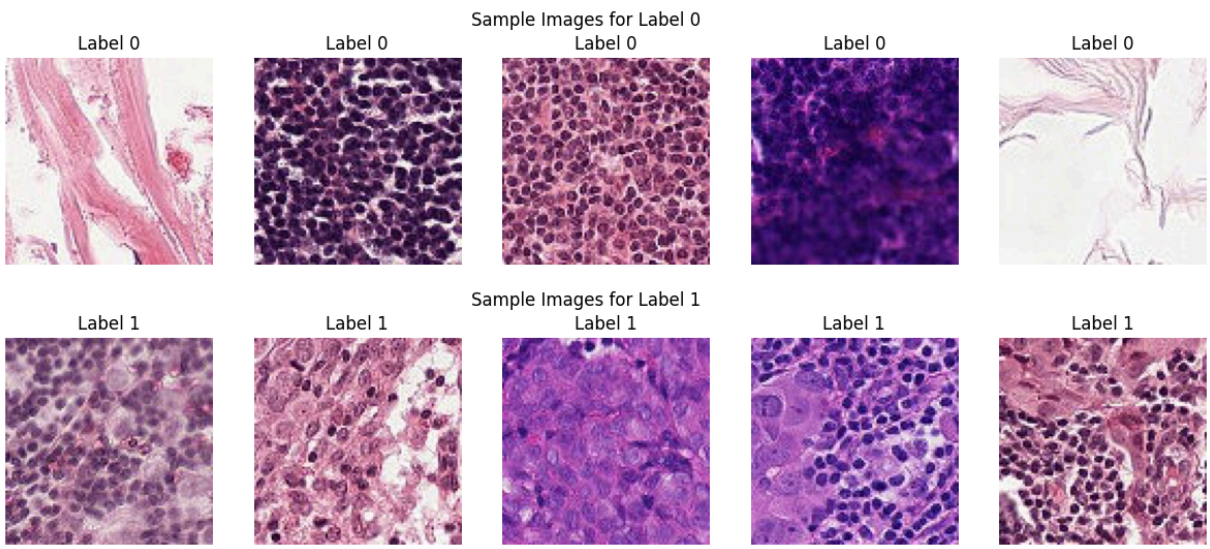
Making a bar chart of counting the labels, seeing how many picture ids are cancerious

In [5]:
```
%matplotlib inline
plt.figure(figsize=(10, 6))
plt.barh(['No Cancer', 'Cancer'], labels_count, color=['#D97D6E', '#6C7A89']
plt.xlabel('Number of Images')
plt.title('Distribution of Labels in Dataset')
plt.show()
```



Looking at the actual data, which images were labeled as cancerous (label 1) and which were labeled as non-cancerous (label 0)

In [6]:
```
def display_sample_images(label, n_samples=5):
    label_samples = labels[labels['label'] == label]
    selected_samples = label_samples.sample(n=n_samples, random_state=42)

    fig, axes = plt.subplots(1, n_samples, figsize=(15, 3))

    for idx, (img_name, ax) in enumerate(zip(selected_samples['id'], axes)):
        img_path = os.path.join(train_dir, f"{img_name}.tif")
        img = Image.open(img_path)

        ax.imshow(img)
        ax.axis('off')

        ax.set_title(f'Label {label}')

    plt.suptitle(f'Sample Images for Label {label}')

    plt.show()

display_sample_images(label=0)
display_sample_images(label=1)
```



Splitting data into training and testing

In [8]:
```
train_labels, val_labels = train_test_split(labels, test_size=0.2, random_st
```

In [10]:
```
train_labels['label'] = train_labels['label'].astype(str)
val_labels['label'] = val_labels['label'].astype(str)
```

Adding .tif at the end becuase that is the correct file name

In [12]:
```
train_labels['filename'] = train_labels['id'] + '.tif'
val_labels['filename'] = val_labels['id'] + '.tif'
```

We use ImageDataGenerator to preprocess the images by rescaling and applying augmentations (flipping) for training, while only rescaling the images for validation.

In [13]:
```
# Image Data Generators
train_datagen = ImageDataGenerator(
    rescale=1./255,
    horizontal_flip=True,
    vertical_flip=True)

val_datagen = ImageDataGenerator(rescale=1./255)

# Train data generator
train_generator = train_datagen.flow_from_dataframe(
    dataframe=train_labels,
    directory=train_dir,
    x_col='filename',
    y_col='label',
    target_size=(96, 96),
    batch_size=32,
    class_mode='binary')

# Validation data generator
val_generator = val_datagen.flow_from_dataframe(
    dataframe=val_labels,
    directory=train_dir,
    x_col='filename',
    y_col='label',
    target_size=(96, 96),
    batch_size=32,
    class_mode='binary')
```

```
Found 176020 validated image filenames belonging to 2 classes.
Found 44005 validated image filenames belonging to 2 classes.
```

Building the CNN

In [14]:
```
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(96, 96, 3)),
    MaxPooling2D(2, 2),

    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2, 2),

    Flatten(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

```
/Users/evelynhaskins/.pyenv/versions/3.10.12/lib/python3.10/site-packages/ke
ras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `
input_shape`/`input_dim` argument to a layer. When using Sequential models,
prefer using an `Input(shape)` object as the first layer in the model instea
d.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Compiling the Model

In [19]:
```
lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=0.001,
    decay_steps=10000,
    decay_rate=0.9,
    staircase=True)

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=lr_schedule),
              loss='binary_crossentropy',
              metrics=['accuracy'])

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Par
conv2d (Conv2D)	(None, 94, 94, 32)	
max_pooling2d (MaxPooling2D)	(None, 47, 47, 32)	
conv2d_1 (Conv2D)	(None, 45, 45, 64)	18
max_pooling2d_1 (MaxPooling2D)	(None, 22, 22, 64)	
flatten (Flatten)	(None, 30976)	
dense (Dense)	(None, 128)	3,965
dense_1 (Dense)	(None, 1)	

Total params: 3,984,577 (15.20 MB)
Trainable params: 3,984,577 (15.20 MB)
Non-trainable params: 0 (0.00 B)

Training the model

In [20]:
```
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    epochs=10,
    validation_data=val_generator,
    validation_steps=val_generator.samples // val_generator.batch_size
)
```

```
Epoch 1/10
5500/5500 ━━━━━━━━━━━━━━━━━━━━ 259s 47ms/step - accuracy: 0.8199 - loss: 0.4
061 - val_accuracy: 0.8581 - val_loss: 0.3311
Epoch 2/10
5500/5500 ━━━━━━━━━━━━━━━━━━━━ 0s 3us/step - accuracy: 0.8750 - loss: 0.2861
- val_accuracy: 0.8000 - val_loss: 0.3610
Epoch 3/10
   3/5500 ━━━━━━━━━━━━━━━━━━━━ 3:42 40ms/step - accuracy: 0.7969 - loss: 0.4
483
```

```
2024-12-04 15:58:10.198114: I tensorflow/core/framework/local_rendezvous.cc:
405] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
         [[{{node IteratorGetNext}}]]
/Users/evelynhaskins/.pyenv/versions/3.10.12/lib/python3.10/contextlib.py:15
3: UserWarning: Your input ran out of data; interrupting training. Make sure
that your dataset or generator can generate at least `steps_per_epoch * epoc
hs` batches. You may need to use the `.repeat()` function when building your
dataset.
  self.gen.throw(typ, value, traceback)
2024-12-04 15:58:10.209327: I tensorflow/core/framework/local_rendezvous.cc:
405] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
         [[{{node IteratorGetNext}}]]
5500/5500 ━━━━━━━━━━━━━━━━━━━━ 255s 46ms/step - accuracy: 0.8578 - loss: 0.3
357 - val_accuracy: 0.8740 - val_loss: 0.3006
Epoch 4/10
5500/5500 ━━━━━━━━━━━━━━━━━━━━ 0s 2us/step - accuracy: 0.9062 - loss: 0.2950
- val_accuracy: 1.0000 - val_loss: 0.0986
Epoch 5/10
   3/5500 ━━━━━━━━━━━━━━━━━━━━ 3:57 43ms/step - accuracy: 0.9028 - loss: 0.3
107
2024-12-04 16:02:25.332265: I tensorflow/core/framework/local_rendezvous.cc:
405] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
         [[{{node IteratorGetNext}}]]
5500/5500 ━━━━━━━━━━━━━━━━━━━━ 250s 45ms/step - accuracy: 0.8729 - loss: 0.3
037 - val_accuracy: 0.8694 - val_loss: 0.3045
Epoch 6/10
5500/5500 ━━━━━━━━━━━━━━━━━━━━ 0s 2us/step - accuracy: 0.8750 - loss: 0.2153
- val_accuracy: 0.6000 - val_loss: 0.5683
Epoch 7/10
5500/5500 ━━━━━━━━━━━━━━━━━━━━ 246s 45ms/step - accuracy: 0.8825 - loss: 0.2
837 - val_accuracy: 0.8897 - val_loss: 0.2675
Epoch 8/10
5500/5500 ━━━━━━━━━━━━━━━━━━━━ 0s 2us/step - accuracy: 0.9375 - loss: 0.2008
- val_accuracy: 1.0000 - val_loss: 0.0454
Epoch 9/10
   3/5500 ━━━━━━━━━━━━━━━━━━━━ 3:55 43ms/step - accuracy: 0.9410 - loss: 0.1
318
2024-12-04 16:10:40.598566: I tensorflow/core/framework/local_rendezvous.cc:
405] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
         [[{{node IteratorGetNext}}]]
5500/5500 ━━━━━━━━━━━━━━━━━━━━ 245s 44ms/step - accuracy: 0.8879 - loss: 0.2
697 - val_accuracy: 0.8872 - val_loss: 0.2919
Epoch 10/10
5500/5500 ━━━━━━━━━━━━━━━━━━━━ 0s 2us/step - accuracy: 0.8750 - loss: 0.3697
- val_accuracy: 1.0000 - val_loss: 0.0504
```

Evaluate the model on the validation set

In [21]:
```
val_loss, val_accuracy = model.evaluate(val_generator)
print(f"Validation Loss: {val_loss}")
print(f"Validation Accuracy: {val_accuracy}")
```

```
1376/1376 ━━━━━━━━━━━━━━━━━━━━ 17s 12ms/step - accuracy: 0.8857 - loss: 0.29
53
Validation Loss: 0.2923514246940613
Validation Accuracy: 0.8867174386978149
```