Chung Yu Ericson Ho

My primary interest for this EDA is:

- Feature Relationships: Analyzing the distribution of continuous magnitude (mag) across categorical factors like network source (net) and the engineered ordinal categories (mag_ordinal, depth_ordinal) to see if the different locations of network would have impact on the data.
- Quantitative Relationships: Quantify the relationships between numerical features the topic is interested in, such as `magnitude`, `depth`, `latitidue`, `longitdue`, `network sorce`, `magnitude ordinal` and `depth ordinal`.
- Spatial Distribution: Mapping earthquake epicenters, particularly identifying regions prone to high-magnitude or deep events.

In [1]:
```
import pandas as pd
import altair as alt
import numpy as np
import altair as alt
alt.renderers.enable("mimetype")
alt.data_transformers.enable("vegafusion")
```

Out[1]:
```
DataTransformerRegistry.enable('vegafusion')
```

In [2]:
```
df = pd.read_csv("../../data/processed/ordinal_data.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	time	latitude	longitude	depth	mag	magType	nst	gap	dmin
0	2025-10-23T22:11:40.587Z	32.274000	-101.931000	4.2122	1.40	ml	40.0	40.0	0.000000
1	2025-10-23T22:09:24.260Z	38.806835	-122.751999	-0.6400	1.27	md	13.0	110.0	0.023310
2	2025-10-23T22:08:01.540Z	38.807835	-122.751167	0.1900	1.24	md	12.0	112.0	0.023690
3	2025-10-23T22:07:48.630Z	38.834332	-122.796333	2.2500	0.23	md	10.0	76.0	0.006201
4	2025-10-23T22:01:31.590Z	38.808998	-122.811668	3.6600	0.74	md	10.0	83.0	0.012830

5 rows × 25 columns

In [4]:
```
df.columns
```

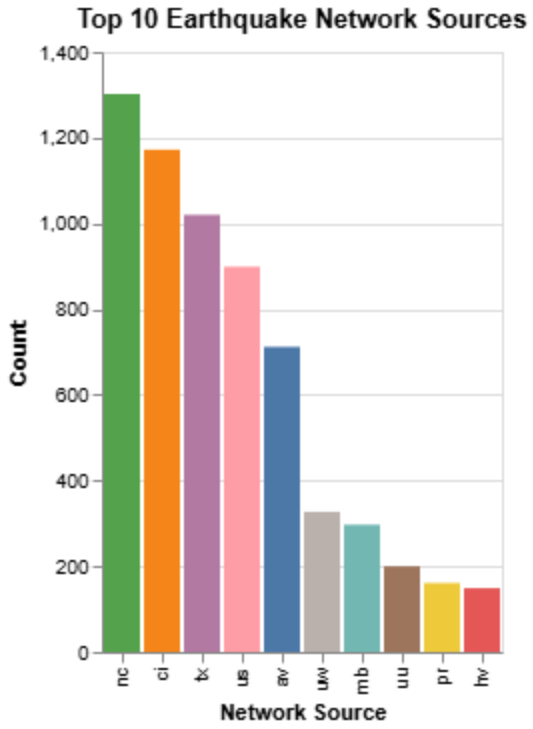
Out[4]:
```
Index(['time', 'latitude', 'longitude', 'depth', 'mag', 'magType', 'nst',
       'gap', 'dmin', 'rms', 'net', 'id', 'updated', 'place', 'type',
       'horizontalError', 'depthError', 'magError', 'magNst', 'status',
       'locationSource', 'magSource', 'mag_ordinal', 'depth_ordinal',
       'gap_level'],
      dtype='object')
```

In [5]:
```
# Define necessary ordinal orders for proper visualization
depth_order = ['Negative (<0 km)', 'Shallow (0-70 km)', 'Intermediate (70-300 km)',
gap_order = ['poor', 'moderate-low', 'moderate-high', 'high']
mag_order = ['Minor (<4.0)', 'Light (4.0-4.9)', 'Moderate (5.0-5.9)', 'Strong (6.0-
core_cols = [
    'mag', 'depth', 'gap', 'latitude', 'longitude'
]
```

Different Network Potential Geographic Impact On Data

In [6]:
```
net_counts = df['net'].value_counts().nlargest(10).index.tolist()
net_bar = alt.Chart(df[df['net'].isin(net_counts)]).mark_bar().encode(
    alt.X("net:N", title="Network Source", sort="-y"),
    alt.Y("count()", title="Count"),
    color=alt.Color("net:N", legend=None)
).properties(title="Top 10 Earthquake Network Sources")
net_bar
```

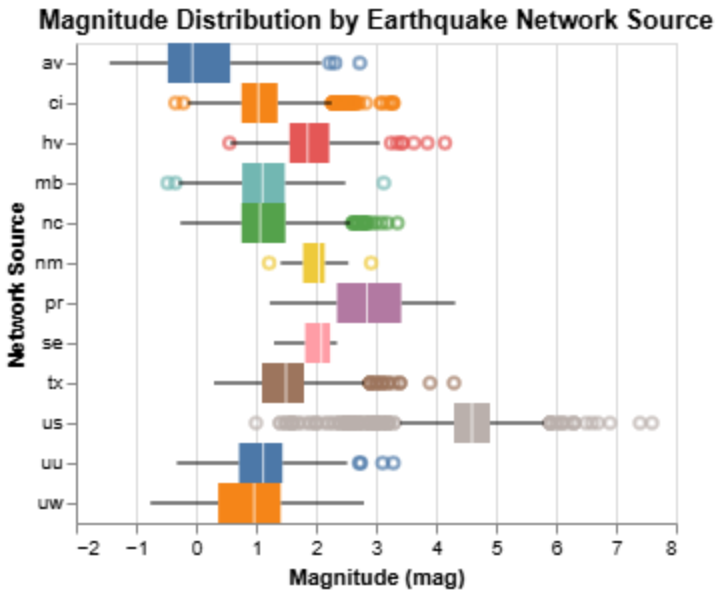
Out[6]:



In [7]:
```
mag_net_boxplot = alt.Chart(df).mark_boxplot(extent=1.5, size=20).encode(
    alt.Y('net:N', title='Network Source', sort=alt.EncodingSortField(field="mag",
    alt.X('mag:Q', title='Magnitude (mag)'),
    alt.Color('net:N', legend=None),
    tooltip=['net', 'median(mag)', 'min(mag)', 'max(mag)']
```

```
).properties(
    title='Magnitude Distribution by Earthquake Network Source'
).interactive()
mag_net_boxplot
```

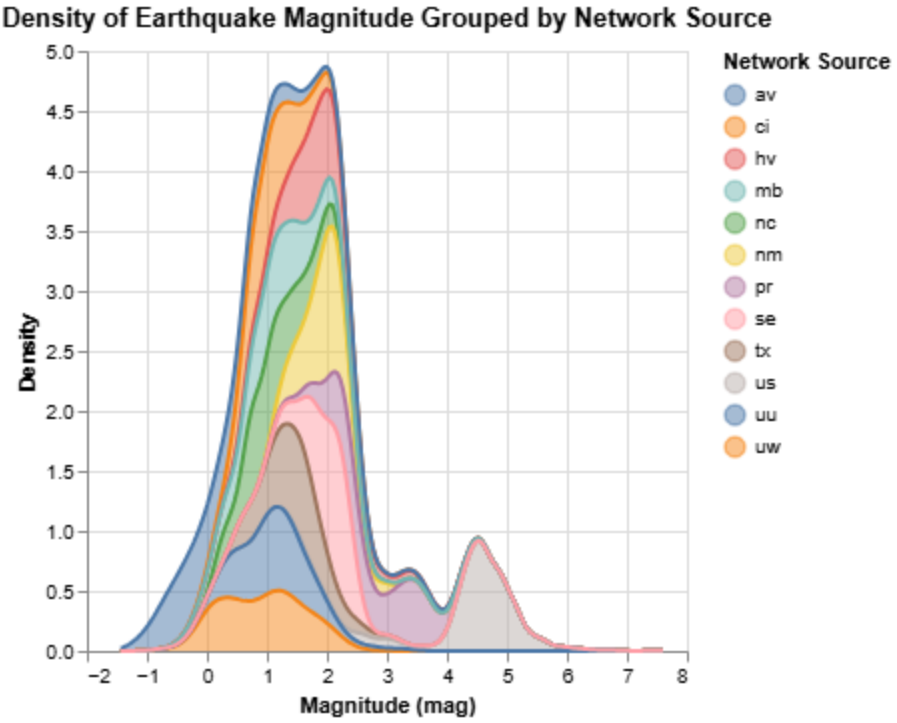
Out[7]:



In [8]:

```
net_density_chart = alt.Chart(df).transform_density(
    'mag',
    groupby=['net'],
    as_=['mag', 'density']
).mark_area(
    opacity=0.5,
    line={'color':'darkgrey'}
).encode(
    x=alt.X('mag:Q', title='Magnitude (mag)'),
    y=alt.Y('density:Q', title='Density'),  # Force quantitative
    color=alt.Color('net:N', title='Network Source'),
    tooltip=[alt.Tooltip('net:N'), alt.Tooltip('mag:Q'), alt.Tooltip('density:Q')]
).properties(
    title='Density of Earthquake Magnitude Grouped by Network Source'
)

net_density_chart
```

Out[8]:

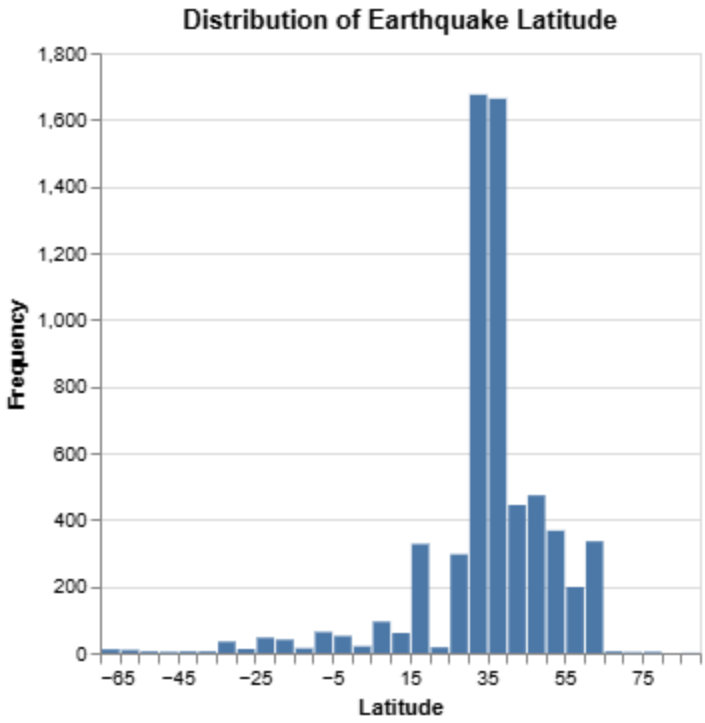


Geolocation of Earthquakes

In [9]:

```
lat_hist = alt.Chart(df).mark_bar().encode(
    alt.X("latitude:Q", bin=alt.Bin(maxbins=50), title="Latitude"),
    alt.Y("count()", title="Frequency")
).properties(title="Distribution of Earthquake Latitude")
lat_hist
```

Out[9]:

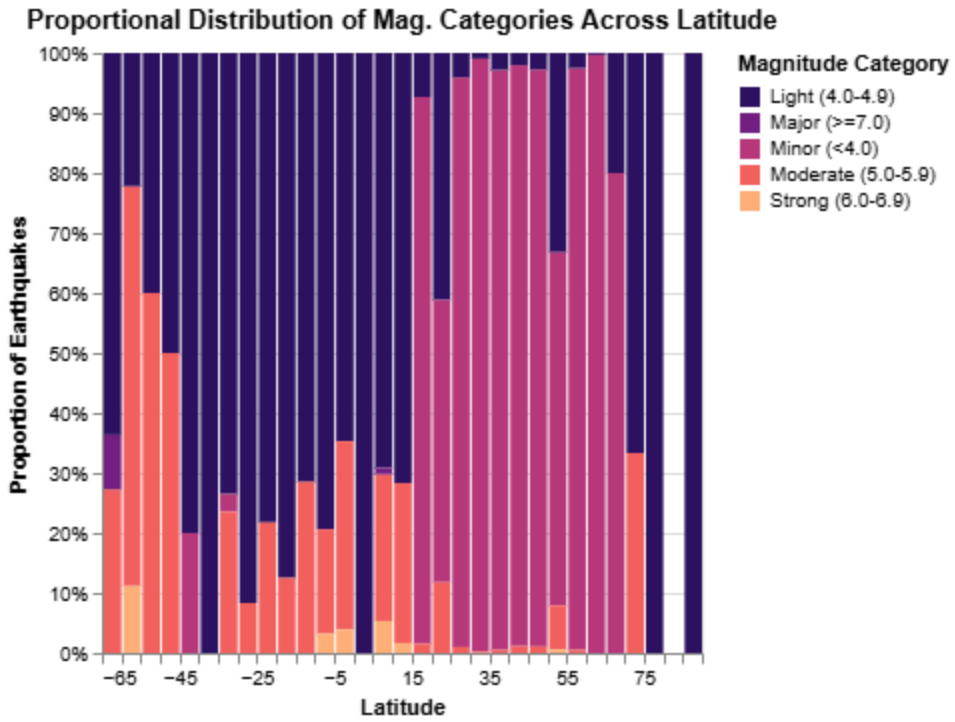


In [10]:

```
lat_proportional_hist = alt.Chart(df).mark_bar(
    strokeWidth=0,
```

```
    stroke=None
).encode(
    alt.X("latitude:Q", bin=alt.Bin(maxbins=30), title="Latitude"),
    alt.Y("count()", stack="normalize", title="Proportion of Earthquakes"),
    alt.Color("mag_ordinal:O", title="Magnitude Category",
              scale=alt.Scale(scheme='magma')),
    tooltip=[
        alt.Tooltip('latitude:Q', bin=True, title='Latitude Bin'),
        alt.Tooltip('mag_ordinal:O', title='Magnitude Category'),
        alt.Tooltip('count():Q', title='Count'),
    ]
).properties(
    title="Proportional Distribution of Mag. Categories Across Latitude"
)

lat_proportional_hist
```

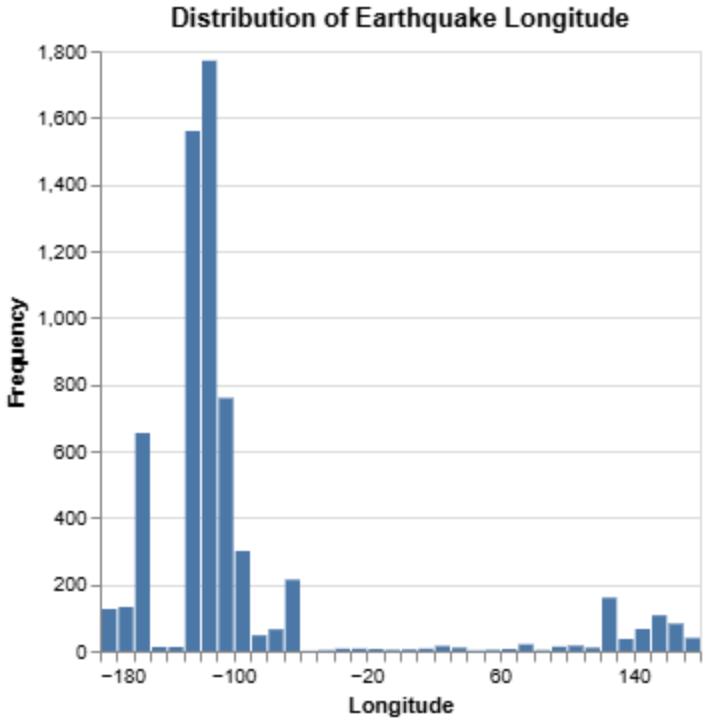
Out[10]:



In [11]:

```
lon_hist = alt.Chart(df).mark_bar().encode(
    alt.X("longitude:Q", bin=alt.Bin(maxbins=50), title="Longitude"),
    alt.Y("count()", title="Frequency")
).properties(title="Distribution of Earthquake Longitude")
lon_hist
```

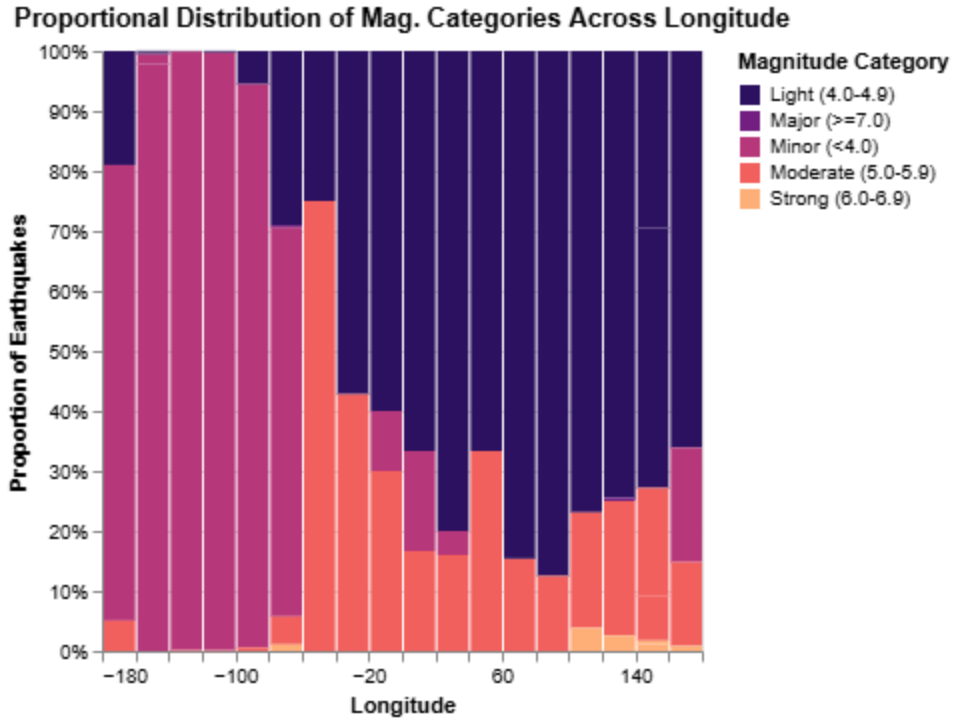
Out[11]:



In [12]:

```
lon_proportional_hist = alt.Chart(df).mark_bar(
    strokeWidth=0,
    stroke=None
).encode(
    alt.X("longitude:Q", bin=alt.Bin(maxbins=30), title="Longitude"),
    alt.Y("count()", stack="normalize", title="Proportion of Earthquakes"),
    alt.Color("mag_ordinal:O", title="Magnitude Category",
              scale=alt.Scale(scheme='magma')),
    tooltip=[
        alt.Tooltip('longitude:Q', bin=True, title='Longitude Bin'),
        alt.Tooltip('mag_ordinal:O', title='Magnitude Category'),
        alt.Tooltip('count():Q', title='Count'),
    ]
).properties(
    title="Proportional Distribution of Mag. Categories Across Longitude"
)

lon_proportional_hist
```

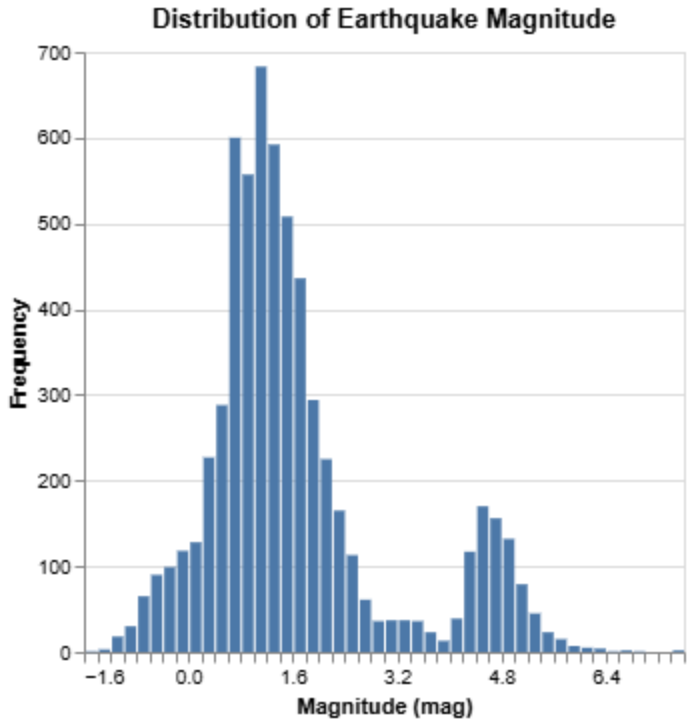
Out[12]:



In [13]:

```
mag_hist = alt.Chart(df).mark_bar().encode(
    alt.X("mag:Q", bin=alt.Bin(maxbins=50), title="Magnitude (mag)"),
    alt.Y("count()", title="Frequency")
).properties(title="Distribution of Earthquake Magnitude")
mag_hist
```

Out[13]:

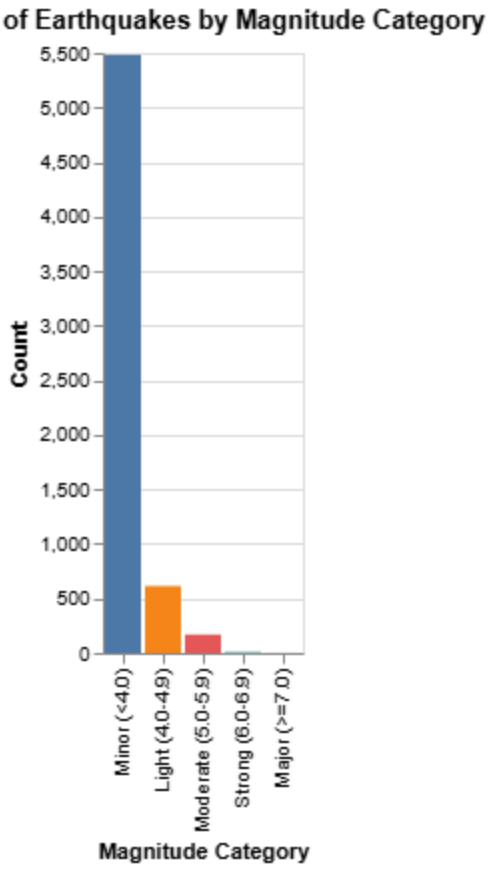


In [14]:

```
mag_ordinal_bar = alt.Chart(df).mark_bar().encode(
    alt.X("mag_ordinal:N", sort=mag_order, title="Magnitude Category"),
    alt.Y("count()", title="Count"),
    color=alt.Color("mag_ordinal:N", sort=mag_order, legend=None)
```

```
).properties(title="Count of Earthquakes by Magnitude Category")
mag_ordinal_bar
```

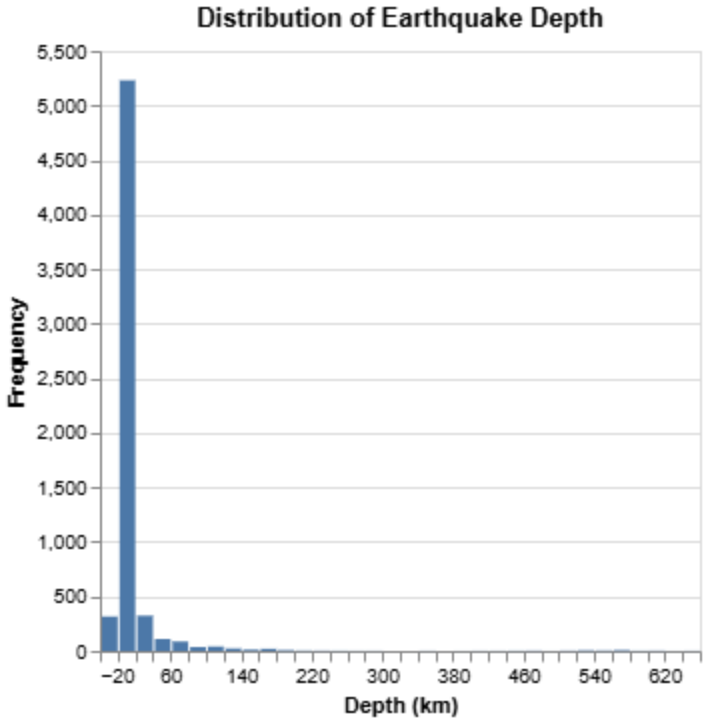
Out[14]:



In [15]:

```
depth_hist = alt.Chart(df).mark_bar().encode(
    alt.X("depth:Q", bin=alt.Bin(maxbins=50), title="Depth (km)"),
    alt.Y("count()", title="Frequency")
).properties(title="Distribution of Earthquake Depth")
depth_hist
```

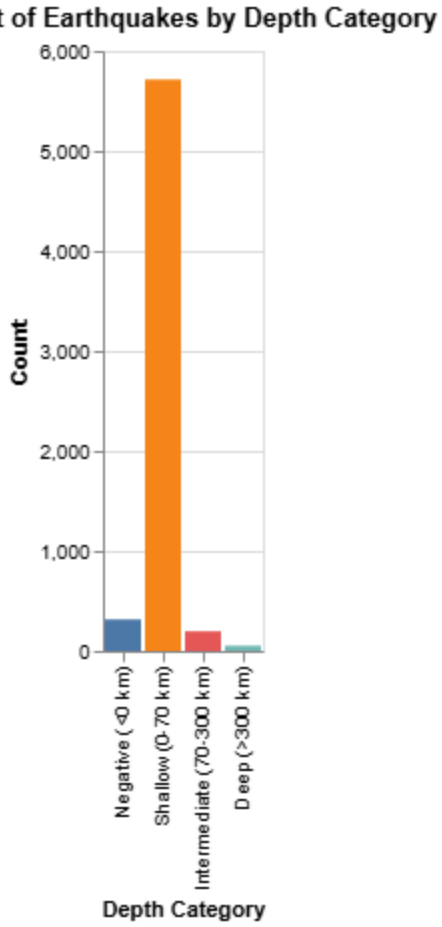
Out[15]:



In [16]:

```
depth_ordinal_bar = alt.Chart(df).mark_bar().encode(
    alt.X("depth_ordinal:N", sort=depth_order, title="Depth Category"),
    alt.Y("count()", title="Count"),
    color=alt.Color("depth_ordinal:N", sort=depth_order, legend=None)
).properties(title="Count of Earthquakes by Depth Category")
depth_ordinal_bar
```

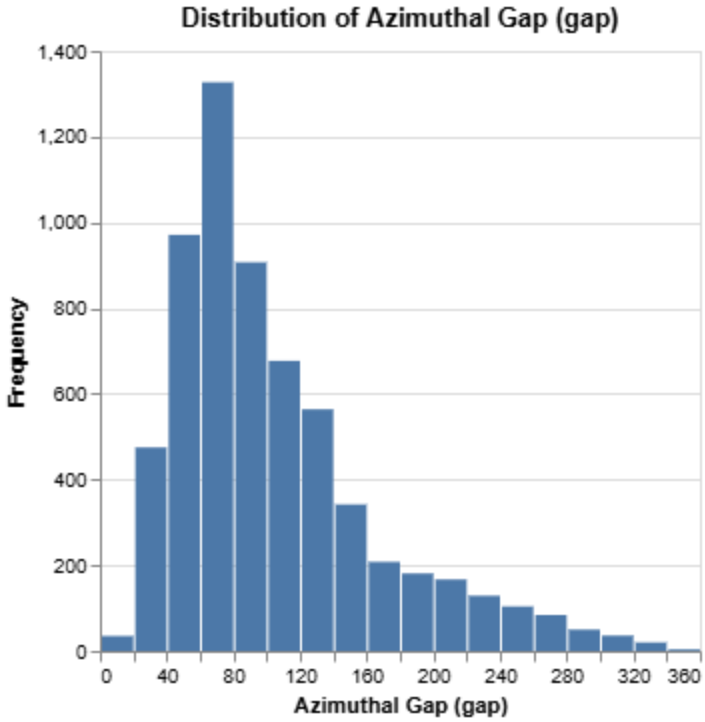
Out[16]:



In [17]:

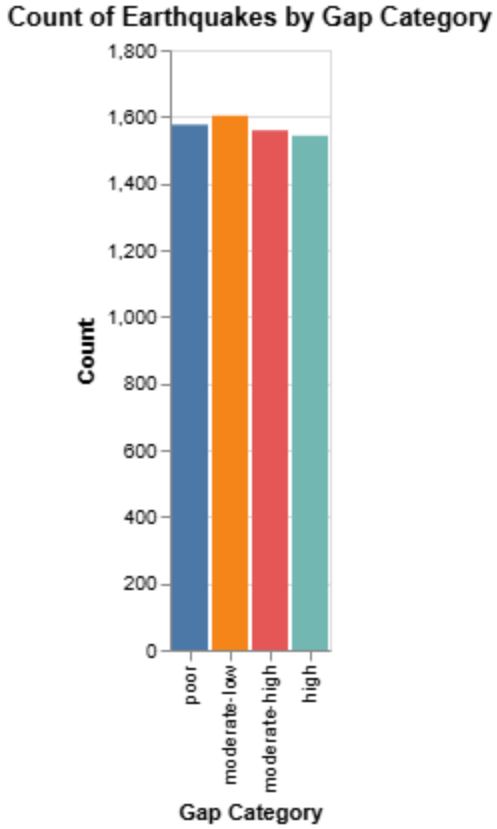
```
gap_hist = alt.Chart(df).mark_bar().encode(
    alt.X("gap:Q", bin=alt.Bin(maxbins=30), title="Azimuthal Gap (gap)"),
    alt.Y("count()", title="Frequency")
).properties(title="Distribution of Azimuthal Gap (gap)")
gap_hist
```

Out[17]:



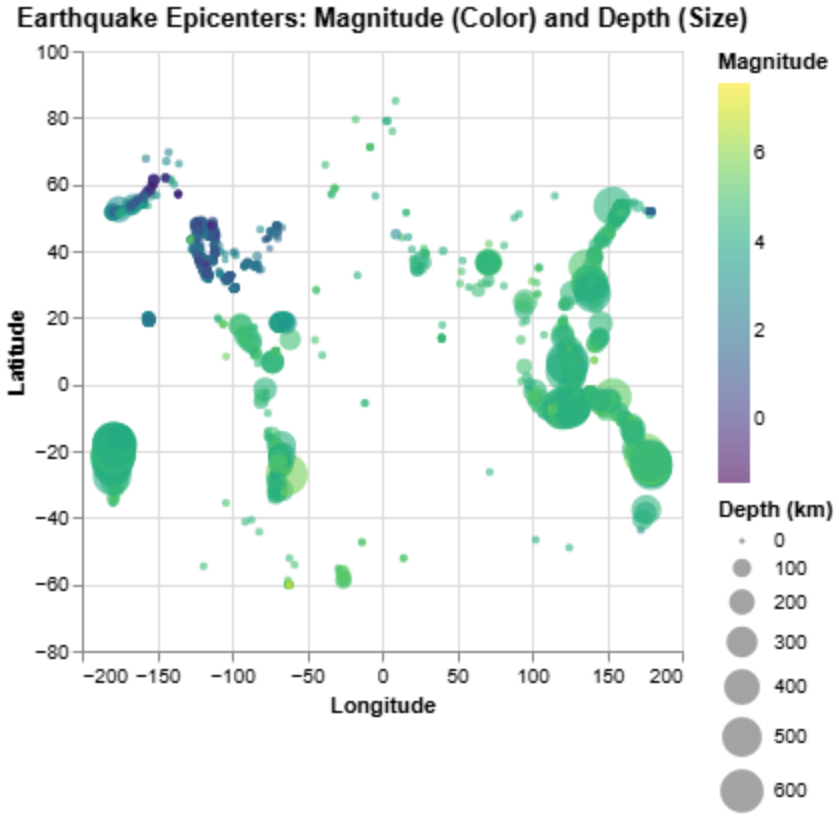
In [18]:
```
depth_ordinal_bar = alt.Chart(df).mark_bar().encode(
    alt.X("gap_level:N", sort=gap_order, title="Gap Category"),
    alt.Y("count()", title="Count"),
    color=alt.Color("gap_level:N", sort=gap_order, legend=None)
).properties(title="Count of Earthquakes by Gap Category")
depth_ordinal_bar
```

Out[18]:



In [19]:
```
spatial_map = alt.Chart(df).mark_circle(opacity=0.6).encode(
    alt.X('longitude:Q', title='Longitude'),
    alt.Y('latitude:Q', title='Latitude'),
    alt.Color('mag:Q', scale=alt.Scale(scheme='viridis', domain=[df['mag'].min(), d
    alt.Size('depth:Q', scale=alt.Scale(range=[5, 500], domain=[df['depth'].min(),
    tooltip=['place', 'mag', 'depth', 'mag_ordinal']
).properties(
    title='Earthquake Epicenters: Magnitude (Color) and Depth (Size)'
)
spatial_map
```

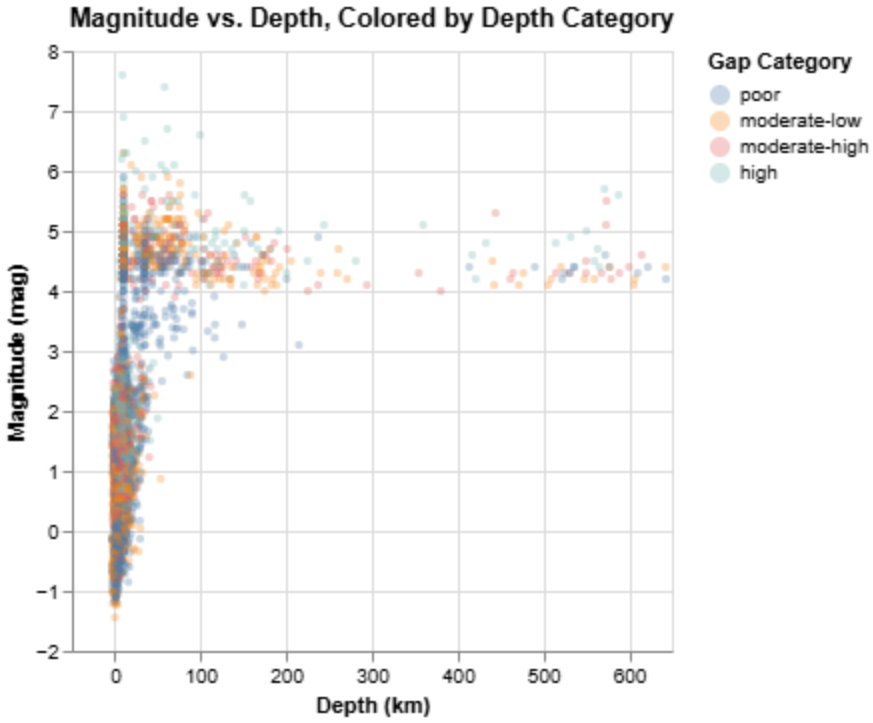
Out[19]:



In [20]:

```
mag_depth_scatter = alt.Chart(df).mark_circle(size=15, opacity=0.3).encode(
    alt.X('depth:Q', title='Depth (km)'),
    alt.Y('mag:Q', title='Magnitude (mag)'),
    alt.Color('gap_level:N', sort=gap_order, title='Gap Category'),
    tooltip=['depth', 'mag', 'depth_ordinal']
).properties(
    title='Magnitude vs. Depth, Colored by Depth Category'
)
mag_depth_scatter
```

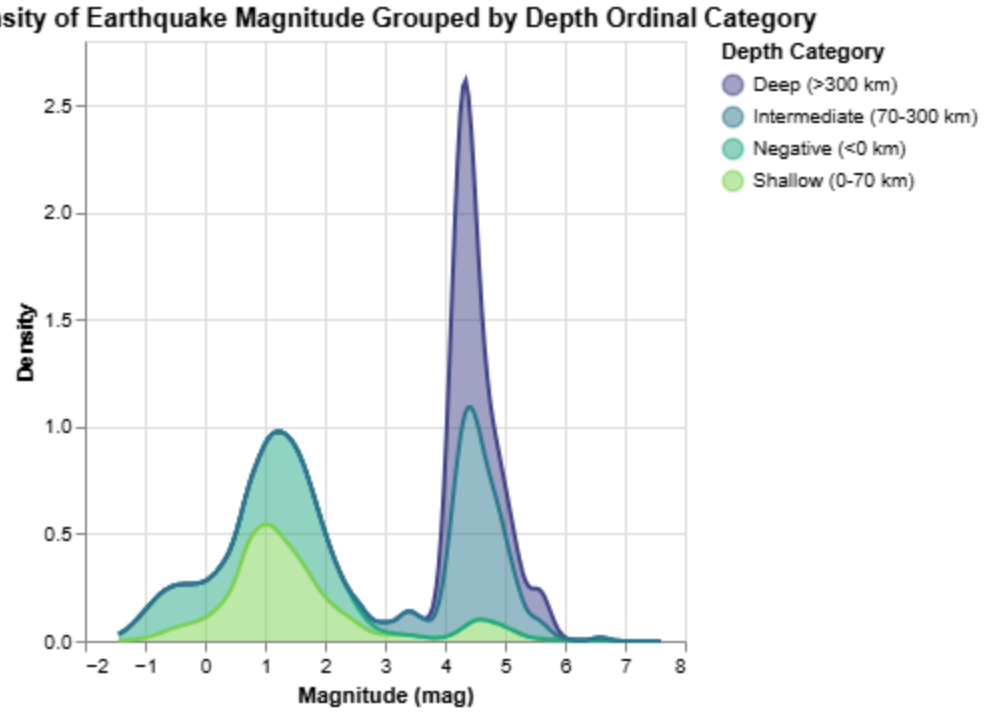
Out[20]:



In [21]:

```
depth_density_chart = alt.Chart(df).transform_density(
    'mag',
    groupby=['depth_ordinal'],
    as_=['mag', 'density']
).mark_area(
    opacity=0.5,
    line={'color':'darkgrey'}
).encode(
    x=alt.X('mag:Q', title='Magnitude (mag)'),
    y=alt.Y('density:Q', title='Density'),

    color=alt.Color('depth_ordinal:N', title='Depth Category', scale=alt.Scale(sche

    tooltip=[
        alt.Tooltip('depth_ordinal:N', title='Depth Category'),
        alt.Tooltip('mag:Q', title='Magnitude'),
        alt.Tooltip('density:Q', title='Density')
    ]
).properties(
    title='Density of Earthquake Magnitude Grouped by Depth Ordinal Category'
)

depth_density_chart
```

Out[21]:



In [22]:

```
corr_matrix_focused = df[core_cols].corr()
cor_data_focused = corr_matrix_focused.stack().reset_index()
cor_data_focused = cor_data_focused.rename(columns={
    0: 'correlation',
    'level_0': 'variable1',
    'level_1': 'variable2'
})
corr_matrix_focused
```

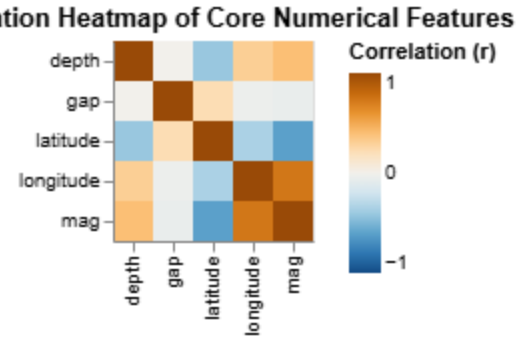
Out[22]:

	mag	depth	gap	latitude	longitude
mag	1.000000	0.397189	-0.056169	-0.607594	0.733623
depth	0.397189	1.000000	0.001089	-0.413894	0.305163
gap	-0.056169	0.001089	1.000000	0.218426	-0.036150
latitude	-0.607594	-0.413894	0.218426	1.000000	-0.348001
longitude	0.733623	0.305163	-0.036150	-0.348001	1.000000

In [23]:

```
cor_data_focused['correlation_label'] = cor_data_focused['correlation'].map('{:.2f}

correlation_heatmap = alt.Chart(cor_data_focused).mark_rect().encode(
    alt.X('variable1:O', title=''),
    alt.Y('variable2:O', title=''),
    alt.Color('correlation', scale=alt.Scale(scheme='blueorange', domain=[-1, 1]),
    text=alt.Text('correlation_label:N'),
    tooltip=['variable1', 'variable2', 'correlation_label']
).properties(
    title='Correlation Heatmap of Core Numerical Features'
).interactive()
correlation_heatmap
```

Out[23]:



Using latitude and longtitude as seperate attribute entry does not give a lot of insight.

In [24]:

```
from vega_datasets import data

world = alt.topo_feature(data.world_110m.url, 'countries')

background_map = alt.Chart(world).mark_geoshape(
    fill='lightgray',
    stroke='white'
).project(
    "equirectangular"
).properties(
    width=700,
    height=400
)
```

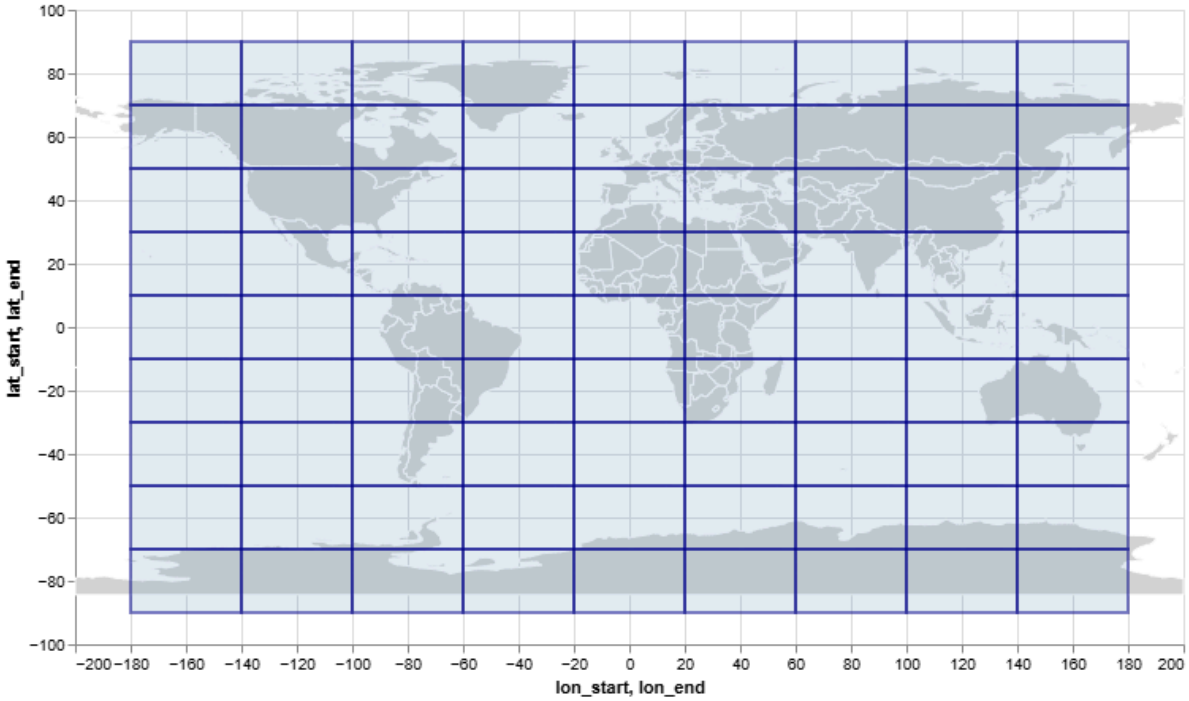
In [25]:

```
import pandas as pd
import numpy as np

lon_step = 40   # wider grid blocks
lat_step = 20

lon_edges = np.arange(-180, 180, lon_step)
lat_edges = np.arange(-90, 90, lat_step)

grid = pd.DataFrame([
    {
        "lon_start": lon,
        "lon_end": lon + lon_step,
        "lat_start": lat,
        "lat_end": lat + lat_step
    }
    for lon in lon_edges
    for lat in lat_edges
])

grid_chart = alt.Chart(grid).mark_rect(
    fill='steelblue',
    fillOpacity=0.15,
    stroke='darkblue',
    strokeWidth=1
).encode(
    x='lon_start:Q',
    x2='lon_end:Q',
```

```
    y='lat_start:Q',
    y2='lat_end:Q',
    tooltip=['lon_start', 'lon_end', 'lat_start', 'lat_end']
).project(
    type="equirectangular"  # same projection as map
)
```

In [26]:
```
map_with_grid = background_map + grid_chart
map_with_grid
```

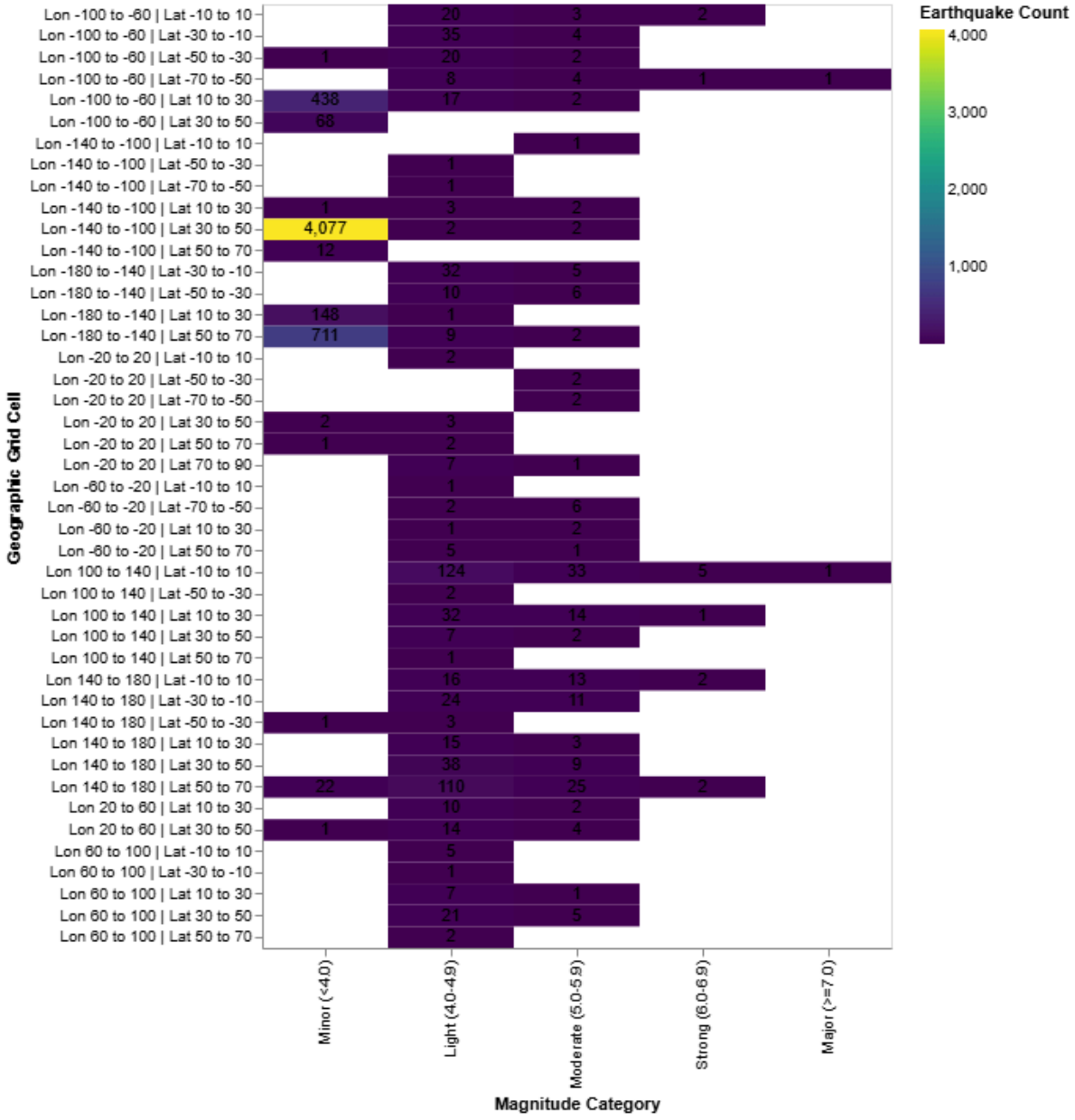
Out[26]:



In [27]:
```
lon_step = 40
lat_step = 20

df['lon_bin'] = np.floor((df['longitude'] + 180) / lon_step) * lon_step - 180
df['lat_bin'] = np.floor((df['latitude'] + 90) / lat_step) * lat_step - 90

df['grid_label'] = (
    'Lon ' + df['lon_bin'].astype(int).astype(str) +
    ' to ' + (df['lon_bin'] + lon_step).astype(int).astype(str) +
    ' | Lat ' + df['lat_bin'].astype(int).astype(str) +
    ' to ' + (df['lat_bin'] + lat_step).astype(int).astype(str)
)

def categorize_magnitude(m):
    if m < 4.0:
        return 'Minor (<4.0)'
    elif m < 5.0:
        return 'Light (4.0-4.9)'
    elif m < 6.0:
        return 'Moderate (5.0-5.9)'
    elif m < 7.0:
        return 'Strong (6.0-6.9)'
    else:
        return 'Major (>=7.0)'
```

```
df['mag_ordinal'] = df['mag'].apply(categorize_magnitude)
mag_order = ['Minor (<4.0)', 'Light (4.0-4.9)', 'Moderate (5.0-5.9)', 'Strong (6.0-

grid_summary = (
    df.groupby(['grid_label', 'mag_ordinal'])
    .size()
    .reset_index(name='count')
)

heatmap = alt.Chart(grid_summary).mark_rect().encode(
    x=alt.X('mag_ordinal:O', sort=mag_order, title='Magnitude Category'),
    y=alt.Y('grid_label:N', title='Geographic Grid Cell'),
    color=alt.Color('count:Q', scale=alt.Scale(scheme='viridis'), title='Earthquake
    tooltip=['grid_label', 'mag_ordinal', alt.Tooltip('count:Q', title='Count')]
)

text = heatmap.mark_text().encode(
    text=alt.Text('count:Q', format=',.0f'),
    color=alt.value('black')
)

final_heatmap_grid = (heatmap + text).properties(width=400, height=600)
final_heatmap_grid
```

Out[27]:



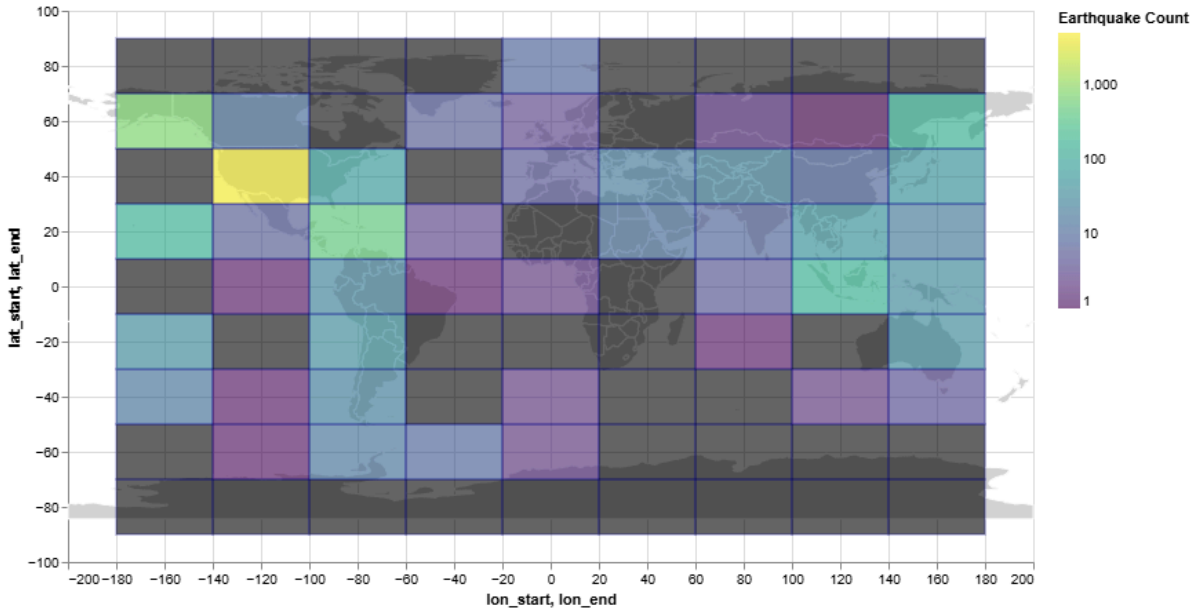
High-Fidelity Sketches Attempts

In [28]:

```
lon_step = 40
lat_step = 20

lon_edges = np.arange(-180, 180, lon_step)
lat_edges = np.arange(-90, 90, lat_step)

grid = pd.DataFrame([
    {
        "lon_start": lon,
        "lon_end": lon + lon_step,
        "lat_start": lat,
        "lat_end": lat + lat_step
    }
    for lon in lon_edges
    for lat in lat_edges
```

```
])

df['lon_bin'] = np.floor((df['longitude'] + 180) / lon_step) * lon_step - 180
df['lat_bin'] = np.floor((df['latitude'] + 90) / lat_step) * lat_step - 90

count_summary = (
    df.groupby(['lon_bin', 'lat_bin'])
    .size()
    .reset_index(name='count')
)

grid_with_counts = pd.merge(
    grid,
    count_summary,
    left_on=['lon_start', 'lat_start'], # These are the bin start values
    right_on=['lon_bin', 'lat_bin'],
    how='left'
).drop(columns=['lon_bin', 'lat_bin'])

grid_with_counts['count'] = grid_with_counts['count'].fillna(0)

geo_heatmap_final = alt.Chart(grid_with_counts).mark_rect(
    stroke='darkblue',
    strokeWidth=0.5,
    opacity=0.6
).encode(
    x='lon_start:Q',
    x2='lon_end:Q',
    y='lat_start:Q',
    y2='lat_end:Q',
    color=alt.Color('count:Q', scale=alt.Scale(scheme='viridis',type="log", domain=
    tooltip=[
        'lon_start', 'lon_end', 'lat_start', 'lat_end',
        alt.Tooltip('count:Q', format=',.0f', title='Count')
    ]
).project(
    type="equirectangular"
)

map_with_heatmap_overlay = background_map + geo_heatmap_final

map_with_heatmap_overlay
```

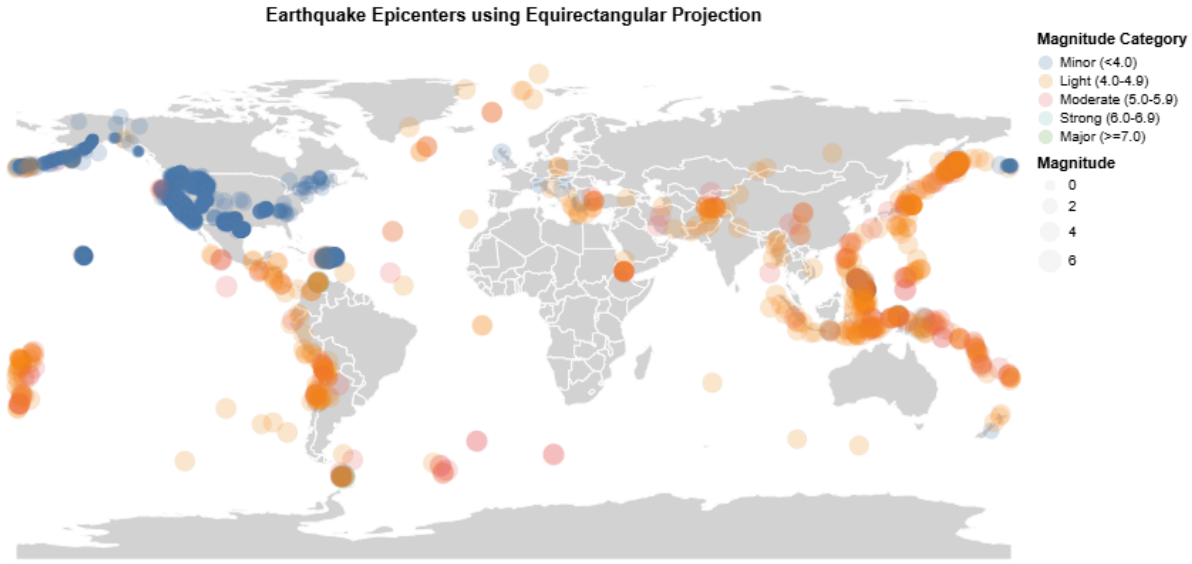
Out[28]:



In [29]:

```
mag_bins = [-np.inf, 4.0, 5.0, 6.0, 7.0, np.inf]
mag_labels = ['Minor (<4.0)', 'Light (4.0-4.9)', 'Moderate (5.0-5.9)', 'Strong (6.0
df['mag_ordinal'] = pd.cut(df['mag'], bins=mag_bins, labels=mag_labels, right=False


mag_order = ['Minor (<4.0)', 'Light (4.0-4.9)', 'Moderate (5.0-5.9)', 'Strong (6.0-


earthquake_layer = alt.Chart(df).mark_circle(opacity=0.2).encode(
    longitude='longitude:Q',
    latitude='latitude:Q',

    color=alt.Color('mag_ordinal:N', sort=mag_order, title='Magnitude Category'),

    size=alt.Size('mag:Q', scale=alt.Scale(range=[0.5, 300]), title='Magnitude'),

    tooltip=['place', 'mag', 'depth', 'mag_ordinal']
)

static_map_final = (background_map + earthquake_layer).properties(
    title='Earthquake Epicenters using Equirectangular Projection'
).interactive()

static_map_final
```

Out[29]:


Earthquake Epicenters using Equirectangular Projection
Magnitude Category
Minor (<4.0)
Light (4.0-4.9)
Moderate (5.0-5.9)
Strong (6.0-6.9)
Major (>=7.0)
Magnitude
0
2
4
6