Installs

Uncomment to install dependencies

In [1]:
```
# !python.exe -m pip install --upgrade pip
# !pip install pandas
# !pip install numpy
# !pip install altair
# !pip install "vegafusion[embed]"
# !pip install vl-convert-python>=1.6.0
# !pip install pyarrow
```

Imports

In [2]:
```
import pandas as pd
import numpy as np
import altair as alt
import vegafusion as vf

alt.data_transformers.enable("vegafusion")
```

Out[2]:
```
DataTransformerRegistry.enable('vegafusion')
```

EDA

Loading the data

In [3]:
```
# Load the dataset and show basic info
df = pd.read_csv("annotations.csv")
# Display basic infor about columns
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 49551 entries, 0 to 49550
Data columns (total 53 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Unnamed: 0                      49551 non-null  int64
 1   person_id                       49551 non-null  int64
 2   filename                        49551 non-null  object
 3   class1                          49551 non-null  object
 4   class2                          3677 non-null   object
 5   bounding_box                    49551 non-null  object
 6   gender_presentation_masc        49551 non-null  int64
 7   gender_presentation_fem         49551 non-null  int64
 8   gender_presentation_non_binary  49551 non-null  int64
 9   gender_presentation_na          49551 non-null  int64
 10  skin_tone_1                     49551 non-null  int64
 11  skin_tone_2                     49551 non-null  int64
 12  skin_tone_3                     49551 non-null  int64
 13  skin_tone_4                     49551 non-null  int64
 14  skin_tone_5                     49551 non-null  int64
 15  skin_tone_6                     49551 non-null  int64
 16  skin_tone_7                     49551 non-null  int64
 17  skin_tone_8                     49551 non-null  int64
 18  skin_tone_9                     49551 non-null  int64
 19  skin_tone_10                    49551 non-null  int64
 20  skin_tone_na                    49551 non-null  int64
 21  age_presentation_young          49551 non-null  int64
 22  age_presentation_older          49551 non-null  int64
 23  age_presentation_middle         49551 non-null  int64
 24  age_presentation_na             49551 non-null  int64
 25  hair_color_brown                49551 non-null  int64
 26  hair_color_blonde               49551 non-null  int64
 27  hair_color_grey                 49551 non-null  int64
 28  hair_color_na                   49551 non-null  int64
 29  hair_color_black                49551 non-null  int64
 30  hair_color_colored              49551 non-null  int64
 31  hair_color_red                  49551 non-null  int64
 32  hairtype_coily                  49551 non-null  int64
 33  hairtype_dreadlocks             49551 non-null  int64
 34  hairtype_bald                   49551 non-null  int64
 35  hairtype_straight               49551 non-null  int64
 36  hairtype_curly                  49551 non-null  int64
 37  hairtype_wavy                   49551 non-null  int64
 38  hairtype_na                     49551 non-null  int64
 39  has_facial_hair                 49551 non-null  int64
 40  has_tattoo                      49551 non-null  int64
 41  has_cap                         49551 non-null  int64
 42  has_mask                        49551 non-null  int64
 43  has_headscarf                   49551 non-null  int64
 44  has_eyeware                     49551 non-null  int64
 45  visible_torso                   49551 non-null  int64
 46  visible_face                    49551 non-null  int64
 47  visible_minimal                 49551 non-null  int64
 48  lighting_underexposed           49551 non-null  int64
 49  lighting_dimly_lit              49551 non-null  int64
 50  lighting_well_lit               49551 non-null  int64
```

```
 51  lighting_na                    49551 non-null  int64
 52  lighting_overexposed           49551 non-null  int64
dtypes: int64(49), object(4)
memory usage: 20.0+ MB
```

Setting the Stage

- Objective: EDA for bias within sensitive attributes such as gender presentation, skin tone presentation, and age presentation.
- Biases: Representation, labeling, sampling, historical, proxy, aggregation
- Proxy candidates: Hair and facial hair

Data Structure Exploration

In [4]:
```
# Show the data
df.describe()
```

Out[4]:

	Unnamed: 0	person_id	gender_presentation_masc	gender_presentation_fem	ge
count	49551.000000	4.955100e+04	49551.000000	49551.000000	
mean	24775.000000	1.307137e+15	0.670824	0.206757	
std	14304.285931	1.293425e+15	0.469919	0.404984	
min	0.000000	3.501556e+14	0.000000	0.000000	
25%	12387.500000	6.396669e+14	0.000000	0.000000	
50%	24775.000000	8.475259e+14	1.000000	0.000000	
75%	37162.500000	1.314718e+15	1.000000	0.000000	
max	49550.000000	9.330696e+15	1.000000	1.000000	

8 rows × 49 columns

Data has already been normalized and follows one-hot encoding for most of its categories. This makes it difficult to relate unless it's done by category counts.

In [5]:
```
# Check for missing values
df.isna().sum() # Only one column has missign entries and it's most cells
```

Out[5]:

Unnamed: 0	0
person_id	0
filename	0
class1	0
class2	45874
bounding_box	0
gender_presentation_masc	0
gender_presentation_fem	0
gender_presentation_non_binary	0
gender_presentation_na	0
skin_tone_1	0
skin_tone_2	0
skin_tone_3	0
skin_tone_4	0
skin_tone_5	0
skin_tone_6	0
skin_tone_7	0
skin_tone_8	0
skin_tone_9	0
skin_tone_10	0
skin_tone_na	0
age_presentation_young	0
age_presentation_older	0
age_presentation_middle	0
age_presentation_na	0
hair_color_brown	0
hair_color_blonde	0
hair_color_grey	0
hair_color_na	0
hair_color_black	0
hair_color_colored	0
hair_color_red	0
hairtype_coily	0
hairtype_dreadlocks	0
hairtype_bald	0
hairtype_straight	0
hairtype_curly	0
hairtype_wavy	0
hairtype_na	0
has_facial_hair	0
has_tattoo	0
has_cap	0
has_mask	0
has_headscarf	0
has_eyeware	0
visible_torso	0
visible_face	0
visible_minimal	0
lighting_underexposed	0
lighting_dimly_lit	0
lighting_well_lit	0
lighting_na	0
lighting_overexposed	0

dtype: int64

Due to how this data is set up, the only missing data entries are in Class 2, which is an optional data category given by META. Also, due to the data being entirely one-hot encoded, there are not really any outliers since most of the data is treated as binary.

In [6]:
```
# Showing the column headers
df.columns
```

Out[6]:
```
Index(['Unnamed: 0', 'person_id', 'filename', 'class1', 'class2',
       'bounding_box', 'gender_presentation_masc', 'gender_presentation_fem',
       'gender_presentation_non_binary', 'gender_presentation_na',
       'skin_tone_1', 'skin_tone_2', 'skin_tone_3', 'skin_tone_4',
       'skin_tone_5', 'skin_tone_6', 'skin_tone_7', 'skin_tone_8',
       'skin_tone_9', 'skin_tone_10', 'skin_tone_na', 'age_presentation_young',
       'age_presentation_older', 'age_presentation_middle',
       'age_presentation_na', 'hair_color_brown', 'hair_color_blonde',
       'hair_color_grey', 'hair_color_na', 'hair_color_black',
       'hair_color_colored', 'hair_color_red', 'hairtype_coily',
       'hairtype_dreadlocks', 'hairtype_bald', 'hairtype_straight',
       'hairtype_curly', 'hairtype_wavy', 'hairtype_na', 'has_facial_hair',
       'has_tattoo', 'has_cap', 'has_mask', 'has_headscarf', 'has_eyeware',
       'visible_torso', 'visible_face', 'visible_minimal',
       'lighting_underexposed', 'lighting_dimly_lit', 'lighting_well_lit',
       'lighting_na', 'lighting_overexposed'],
      dtype='object')
```

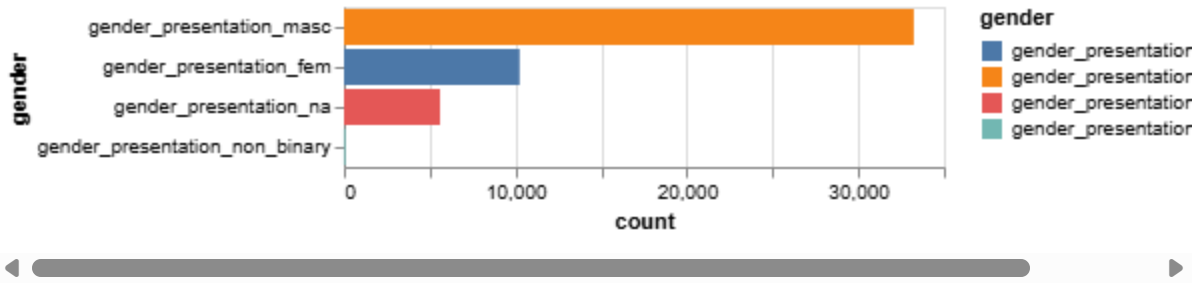
In [7]:
```
# Grouping gender together as an indexable list for altair
gender_cols = [
    "gender_presentation_masc",
    "gender_presentation_fem",
    "gender_presentation_non_binary",
    "gender_presentation_na"
]
# Displaying the number of each type of presenting individuals
GenderCounts = df[gender_cols].sum().reset_index(name="count").rename(columns={"ind
GenderCounts
```

Out[7]:

	gender	count
0	gender_presentation_masc	33240
1	gender_presentation_fem	10245
2	gender_presentation_non_binary	95
3	gender_presentation_na	5602

In [8]:
```
# Using altair to graph the number of gender presentations (Sorted)
alt.Chart(GenderCounts).mark_bar().encode(
    x="count:Q",
    y= alt.Y("gender:N").sort('-x'),
    color="gender:N"
)
```

Out[8]:

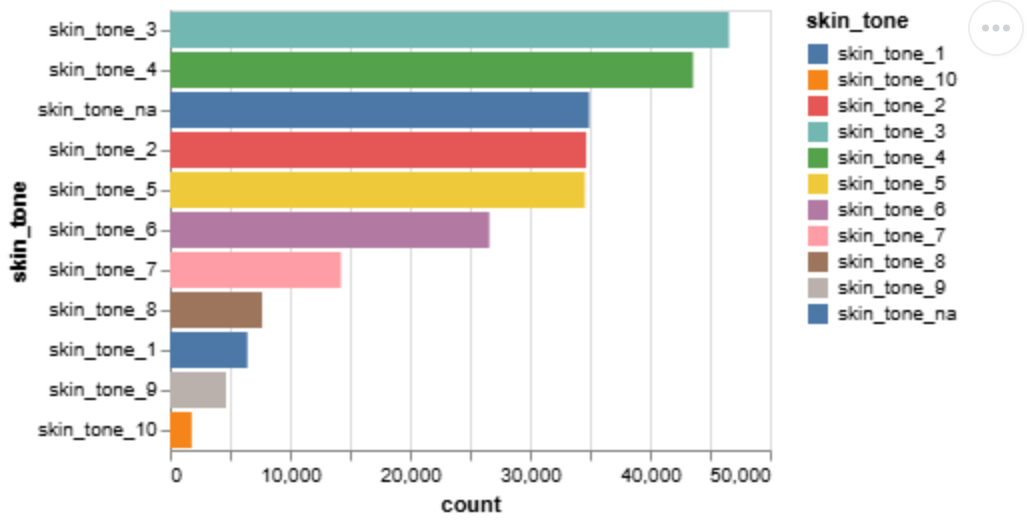


This data is heavily masculine-presenting. There are 3x times as many identified masculine individuals as feminine individuals.

In [9]:

```
# Isolating the skintones
skin_cols = []
for column in df.columns:
    if column.startswith("skin_tone_"):
        skin_cols.append(column)

# Counting the number of different skin tones represented
skin_counts = df[skin_cols].sum().reset_index(name="count").rename(columns={"index"

# Displaying the number of different skin tones represented
alt.Chart(skin_counts).mark_bar().encode(
    x="count:Q",
    y= alt.Y("skin_tone:N").sort('-x'),
    color="skin_tone:N"
)
```

Out[9]:



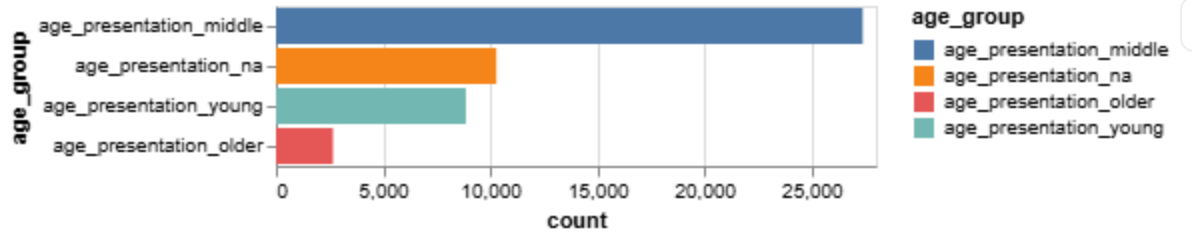
Darker skin tones are less represented than lighter skin tones. However, the mid-range of skin tones has the most representation.

In [10]:

```
# Isolating the ages
age_cols = []
for column in df.columns:
    if column.startswith("age_presentation_"):
        age_cols.append(column)

# Counting the number of different ages represented
```

```
age_counts = df[age_cols].sum().reset_index(name="count").rename(columns={"index":

# Displaying the number of different ages represented
alt.Chart(age_counts).mark_bar().encode(
    x="count:Q",
    y= alt.Y("age_group:N").sort('-x'),
    color="age_group:N"
)
```

Out[10]:

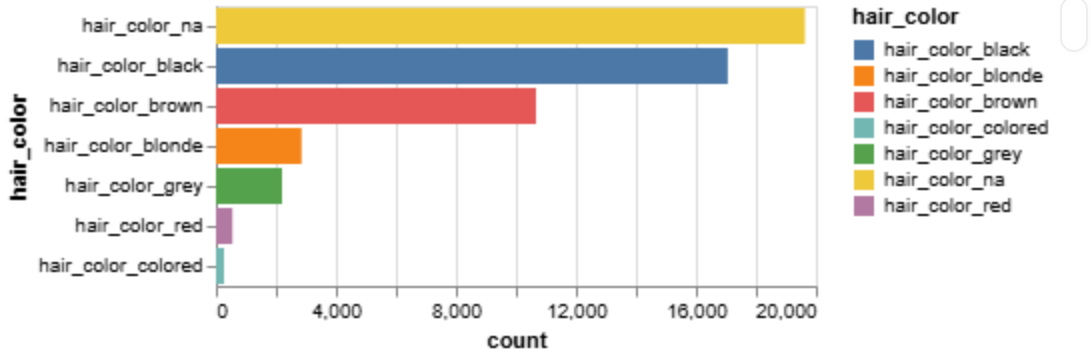


While not as aggressive as gender identification, individuals in the middle age range are overrepresented compared to young or old individuals.

In [11]:

```
# Isolating the hair color
hair_cols = []
for column in df.columns:
    if column.startswith("hair_color_"):
        hair_cols.append(column)

# Counting the number of different hairs colors represented
hair_counts = df[hair_cols].sum().reset_index(name="count").rename(columns={"index"

# Displaying the number of different hairs colors represented
alt.Chart(hair_counts).mark_bar().encode(
    x="count:Q",
    y= alt.Y("hair_color:N").sort('-x'),
    color="hair_color:N"
)
```

Out[11]:



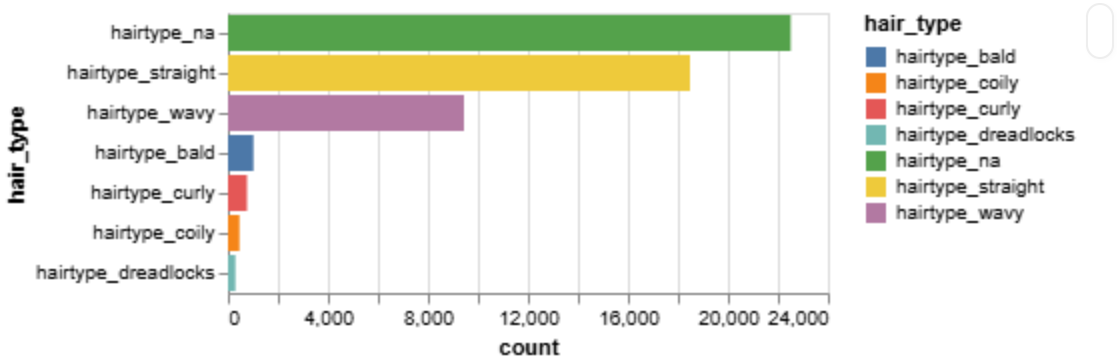
Hair color distributions seem normal enough, though a good chunk of hair types are unidentified.

In [12]:

```
# Isolating the hair color
hair_type_cols = []
for column in df.columns:
```

```
    if column.startswith("hairtype_"):
        hair_type_cols.append(column)

# Counting the number of different hairs colors represented
hair_type_counts = df[hair_type_cols].sum().reset_index(name="count").rename(column

# Displaying the number of different hairs colors represented
alt.Chart(hair_type_counts).mark_bar().encode(
    x="count:Q",
    y= alt.Y("hair_type:N").sort('-x'),
    color="hair_type:N"
)
```

Out[12]:

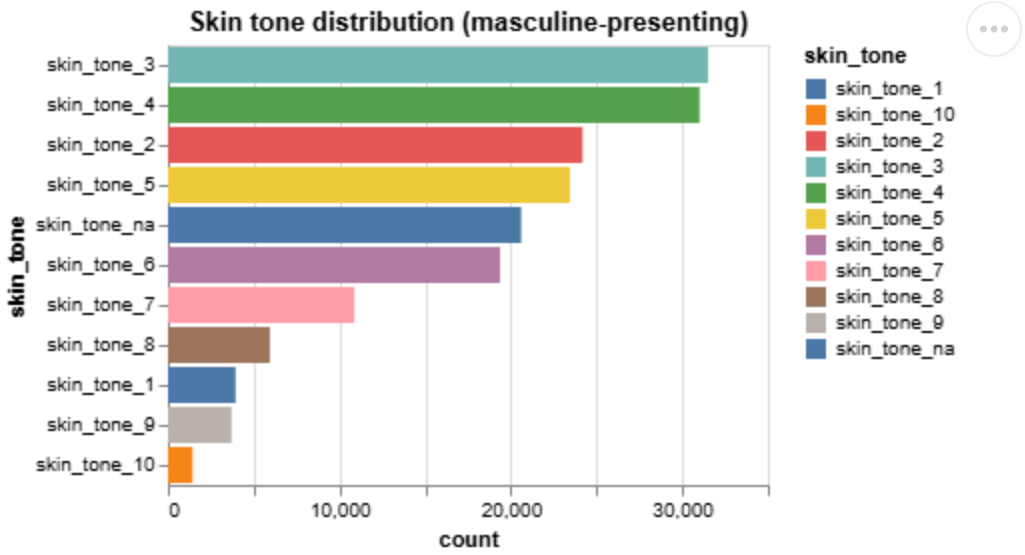


Hair type follows closely with hair color.

In [13]:

```
# Gender masc + skin tone distribution
# Collecting all instances of masc presenting individuals
subset = df[df["gender_presentation_masc"] == 1]
# Counting the number of masc presenting individuals by skin tone
skin_counts_masc = subset[skin_cols].sum().reset_index(name="count").rename(columns

# Displaing number of different skin tones in the masc presenting group
alt.Chart(skin_counts_masc).mark_bar().encode(
    x="count:Q",
    y=alt.Y("skin_tone:N").sort('-x'),
    color="skin_tone:N"
).properties(title="Skin tone distribution (masculine-presenting)")
```

Out[13]:

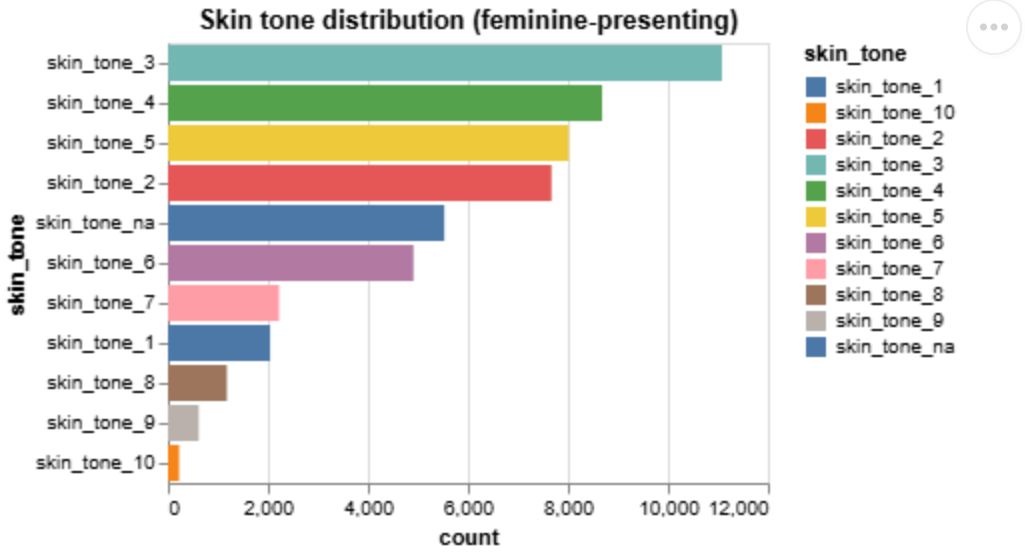


Lighter skintones are far more present in the data set than darker skintones. Lower-numbered skin tones are lighter and higher-numbered skin tones are darker. 1 is the lightest and 10 is the darkest. Of the Masc presenting individuals, FACET contains the most identification of individuals from the 2-5 range. The skin tones can be found here: https://skintone.google/get-started

In [14]:

```
# Gender fem + skin tone distribution
# Collecting all instances of masc presenting individuals
subset = df[df["gender_presentation_fem"] == 1]
# Counting the number of masc presenting individuals by skin tone
skin_counts_fem = subset[skin_cols].sum().reset_index(name="count").rename(columns=

# Displaing number of different skin tones in the masc presenting group
alt.Chart(skin_counts_fem).mark_bar().encode(
    x="count:Q",
    y=alt.Y("skin_tone:N").sort('-x'),
    color="skin_tone:N"
).properties(title="Skin tone distribution (feminine-presenting)")
```

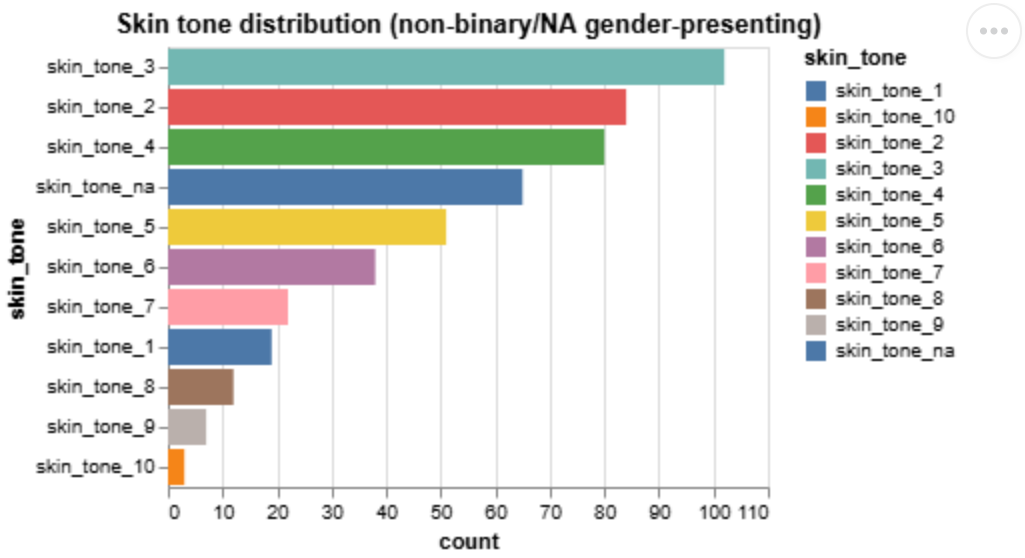
Out[14]:



The Fem presenting individuals also follow a similar distribution to the masc presenting individuals, although there are fewer counts due to a lack of fem presenting data points.

In [15]:
```
# Filter for non-binary and NA gender presentation
subset_nb_na = df[(df["gender_presentation_non_binary"] == 1)]

# Count skin tones for this subset
skin_counts_nb_na = subset_nb_na[skin_cols].sum().reset_index(name="count").rename(

# Plot
alt.Chart(skin_counts_nb_na).mark_bar().encode(
    x="count:Q",
    y=alt.Y("skin_tone:N").sort('-x'),
    color="skin_tone:N"
).properties(title="Skin tone distribution (non-binary/NA gender-presenting)")
```

Out[15]:



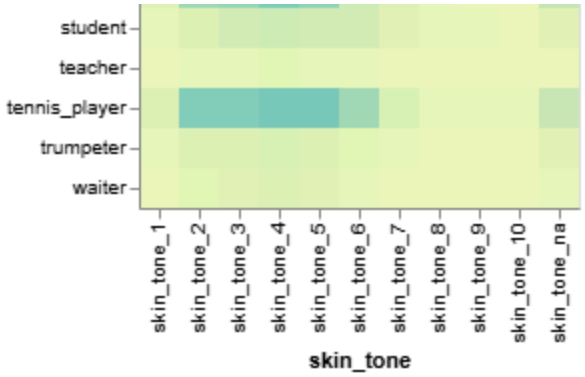
Even among those who present as non-binary, we get a similar distribution. This could show failure on the part of the data collection side to get equal amounts of different skin tones amongst participants. Likely an indication of historical or societal bias.

In [16]:
```
# Melt the skin tone columns to long format
skin_long = df.melt(
    id_vars=["class1"],
    value_vars=skin_cols,
    var_name="skin_tone",
    value_name="present"
)
# Only keep rows where present == 1
skin_long = skin_long[skin_long["present"] == 1]

heatmap_data = (
    skin_long.groupby(["class1", "skin_tone"]).size().reset_index(name="count")
)

# Sort skin_tone categories from least to greatest (1-10, na last)
skin_tone_order = [f"skin_tone_{i}" for i in range(1, 11)] + ["skin_tone_na"]
```

```
alt.Chart(heatmap_data).mark_rect().encode(
    x=alt.X("skin_tone:N", sort=skin_tone_order),
    y="class1:N",
    color="count:Q"
).properties(title="Heatmap of class1 x Skin Tone (sorted)")
```

Out[16]:

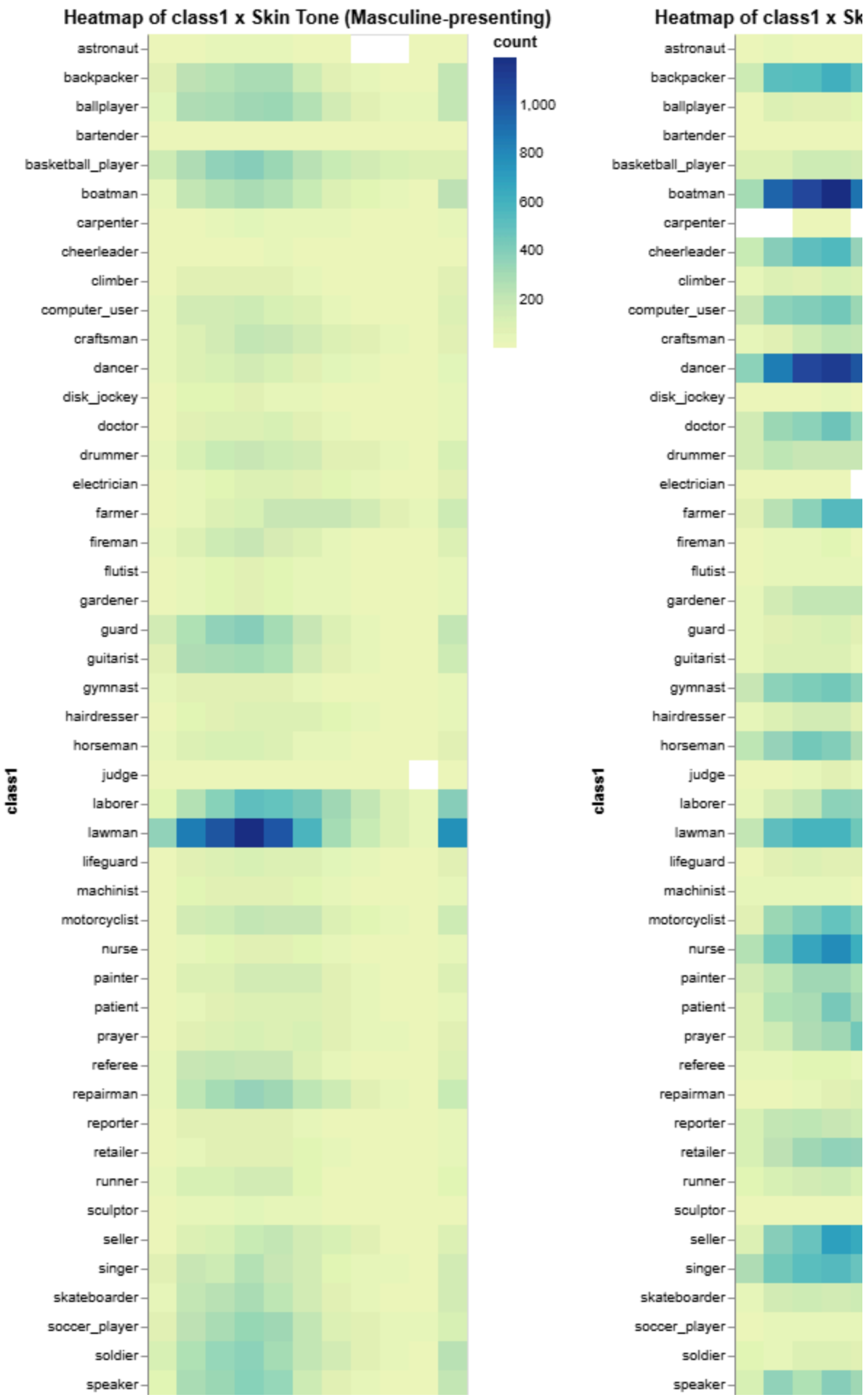

Heatmap of class1 x Skin Tone (sorted)
count
1,449
1
class1
astronaut
backpacker
ballplayer
bartender
basketball_player
boatman
carpenter
cheerleader
climber
computer_user
craftsman
dancer
disk_jockey
doctor
drummer
electrician
farmer
fireman
flutist
gardener
guard
guitarist
gymnast
hairdresser
horseman
judge
laborer
lawman
lifeguard
machinist
motorcyclist
nurse
painter
patient
prayer
referee
repairman
reporter
retailer
runner
sculptor
seller
singer
skateboarder
soccer_player
soldier
speaker

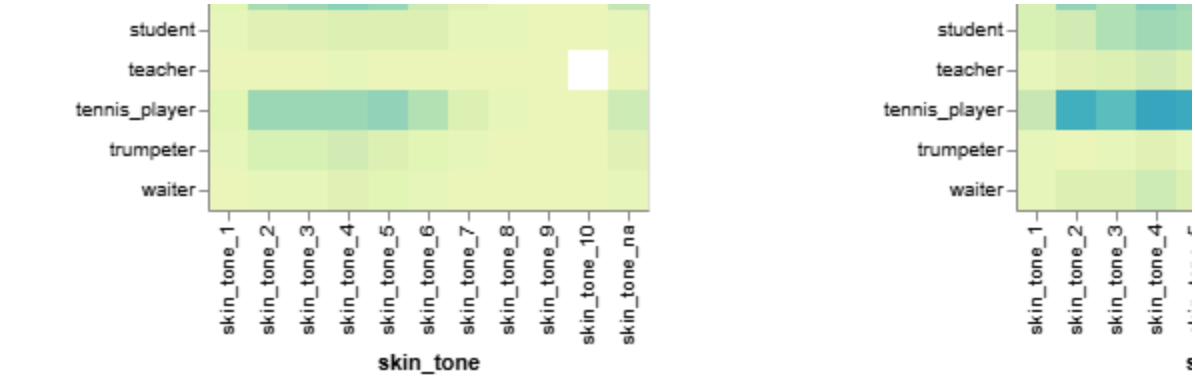


The heat map reveals some intresting characteristics about how the data is classified however I belive that the gendered nature of the data is skewing results for the feminine presenting individuals.

In [17]:
```
# Heatmap for masculine-presenting individuals
skin_long_masc = df[df["gender_presentation_masc"] == 1].melt(
    id_vars=["class1"],
    value_vars=skin_cols,
    var_name="skin_tone",
    value_name="present"
)
skin_long_masc = skin_long_masc[skin_long_masc["present"] == 1]
heatmap_data_masc = (
    skin_long_masc.groupby(["class1", "skin_tone"]).size().reset_index(name="count"
)
chart_masc = alt.Chart(heatmap_data_masc).mark_rect().encode(
    x=alt.X("skin_tone:N", sort=skin_tone_order),
    y="class1:N",
    color="count:Q"
).properties(title="Heatmap of class1 x Skin Tone (Masculine-presenting)")

# Heatmap for feminine-presenting individuals
skin_long_fem = df[df["gender_presentation_fem"] == 1].melt(
    id_vars=["class1"],
    value_vars=skin_cols,
    var_name="skin_tone",
    value_name="present"
)
skin_long_fem = skin_long_fem[skin_long_fem["present"] == 1]
heatmap_data_fem = (
    skin_long_fem.groupby(["class1", "skin_tone"]).size().reset_index(name="count")
)
chart_fem = alt.Chart(heatmap_data_fem).mark_rect().encode(
    x=alt.X("skin_tone:N", sort=skin_tone_order),
    y="class1:N",
    color="count:Q"
).properties(title="Heatmap of class1 x Skin Tone (Feminine-presenting)")

#printing the graphs next to each other
(chart_masc | chart_fem).resolve_scale(color='independent')
```

Out[17]:


Heatmap of class1 x Skin Tone (Masculine-presenting)
count
1,000
800
600
400
200
class1
astronaut
backpacker
ballplayer
bartender
basketball_player
boatman
carpenter
cheerleader
climber
computer_user
craftsman
dancer
disk_jockey
doctor
drummer
electrician
farmer
fireman
flutist
gardener
guard
guitarist
gymnast
hairdresser
horseman
judge
laborer
lawman
lifeguard
machinist
motorcyclist
nurse
painter
patient
prayer
referee
repairman
reporter
retailer
runner
sculptor
seller
singer
skateboarder
soccer_player
soldier
speaker
Heatmap of class1 x Sk
class1

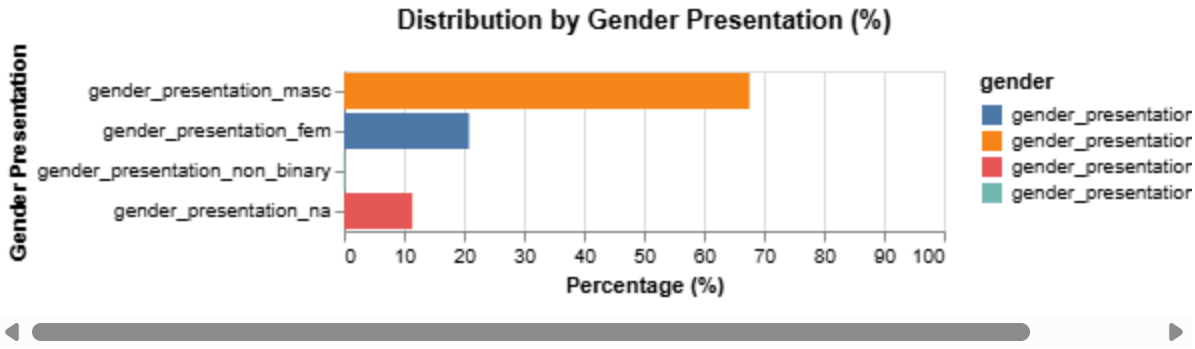


The above graphs are demographic breakdowns by skin tone and class. Essentially, what it is showing is which skintones most often become associated with which classifications. One of the more startling pieces of information these graphs reveal is that darker-skinned women aren't even classified as certain jobs. Things like astronaut and electrician are completely void of women with skintones darker than 5.

Disaggregate by Sensitive Attributes

In [18]:
```
# Calculate percentages for gender
GenderCounts_pct = GenderCounts.copy()
GenderCounts_pct["percent"] = (GenderCounts_pct["count"] / GenderCounts_pct["count"

# Calculate percentages for skin tone
skin_counts_pct = skin_counts.copy()
skin_counts_pct["percent"] = (skin_counts_pct["count"] / skin_counts_pct["count"].s

# Gender bar chart (percentages, normalized to 100%)
gender_chart_pct = alt.Chart(GenderCounts_pct).mark_bar().encode(
    x=alt.Y("percent:Q", title="Percentage (%)", scale=alt.Scale(domain=[0, 100])),
    y=alt.X("gender:N", title="Gender Presentation", sort=gender_cols),
    color="gender:N"
).properties(title="Distribution by Gender Presentation (%)")

# Skin tone bar chart (percentages, normalized to 100%)
skin_chart_pct = alt.Chart(skin_counts_pct).mark_bar().encode(
    x=alt.Y("percent:Q", title="Percentage (%)", scale=alt.Scale(domain=[0, 100])),
    y=alt.X("skin_tone:N", title="Skin Tone", sort=skin_tone_order),
    color="skin_tone:N"
).properties(title="Distribution by Skin Tone (%)")

#Gender
gender_chart_pct
```

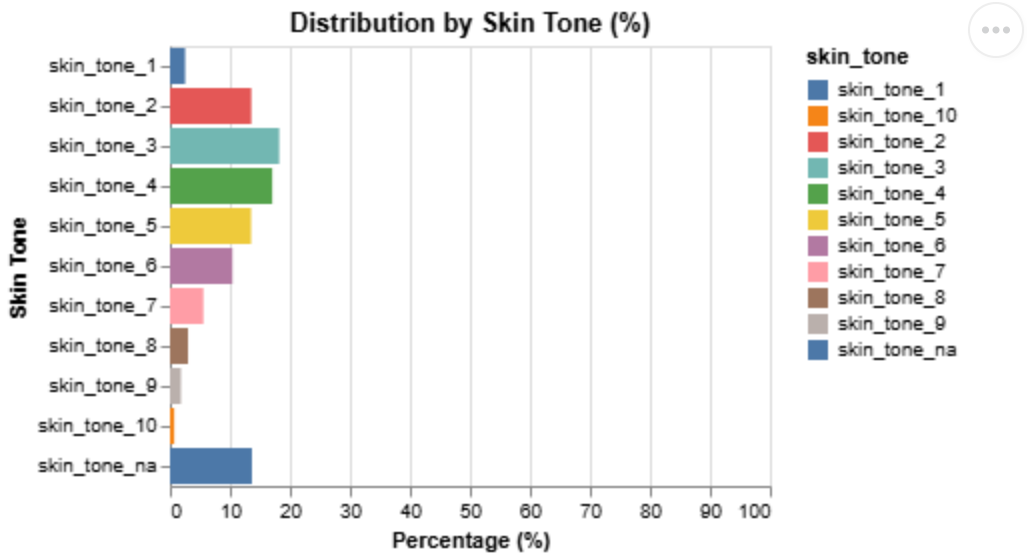
Out[18]:



Masc presenting indivuals once again make up most of the data. Non binary representation is virtually zero.

In [19]:
```
#Skin
skin_chart_pct
```

Out[19]:



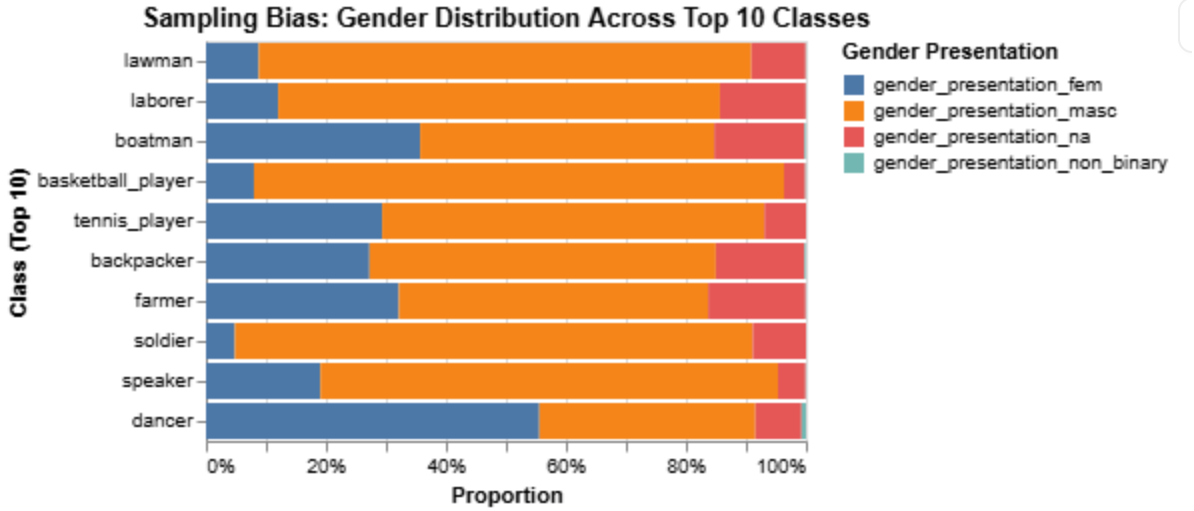
More proof that indviduals with darker skin tones are not well represented.

Sampling Bias

In [20]:
```
# Visualize sampling bias by comparing class (occupation) distribution across gende
# Melt gender columns to long format for class1
gender_long = df.melt(
    id_vars=["class1"],
    value_vars=gender_cols,
    var_name="gender",
    value_name="present"
)
gender_long = gender_long[gender_long["present"] == 1]

# Count number of samples per class1 and gender
class_gender_counts = (
    gender_long.groupby(["class1", "gender"]).size().reset_index(name="count")
)
```

```
# Show top 10 most common classes by total count
top_classes = (
    class_gender_counts.groupby("class1")["count"].sum()
    .sort_values(ascending=False)
    .head(10)
    .index.tolist()
)
class_gender_top = class_gender_counts[class_gender_counts["class1"].isin(top_class

# Plot: Stacked bar chart of gender distribution for top classes
alt.Chart(class_gender_top).mark_bar().encode(
    y=alt.X("class1:N", sort=top_classes, title="Class (Top 10)"),
    x=alt.Y("count:Q", stack="normalize", title="Proportion"),
    color=alt.Color("gender:N", title="Gender Presentation")
).properties(
    title="Sampling Bias: Gender Distribution Across Top 10 Classes"
)
```

Out[20]:



While this isn't a comphrensive list of all the classes, amongst the top ten class the gender presentation make up is heavily skewed to be men except for in the case of the dancer. It's mostly comprised of women which is likely indicitive of gender bias through historical and social contexts.

In total it seems that this data set suffers heavily from gender bias, sampling bais, and racial bias. Out of around 50k entries 70% of them are masc presenting. With in the data classifications it also shows that fem presenting individuals are not represented in all catagories espcially not those that have darker skin. Individuals with darker skin are not nearly as prevelent in the data as those with lighter skin. This inharently will create some kind of issue when it comes to training a model. The most likely option is that it will fail to correctly identify indiviudals with darker skin when trying to classify them.

Dataset Documentation

Dataset Overview

Name: Lucas Drager & Jack Manning

Owner/Contact: [Class project team / instructor]

Access: Class use only, internal distribution

Contents:

- **Items:** 49,551 annotated entries (person/image bounding boxes).
- **Data fields:**
  - Demographics: gender presentation (masc, fem, non-binary, NA), age presentation (young, middle, older, NA), skin tone (10-scale, NA).
  - Appearance: hair color, hair type, presence of facial hair, tattoos, masks, headscarves, eyewear.
  - Visual conditions: visibility of face/torso, lighting quality.
  - Metadata: bounding box coordinates in JSON format, file name, class labels (e.g., “doctor,” “speaker”).
- **Timeframe:** Static snapshot; timeframe not explicitly defined.

Intended Uses:

- Bias and fairness analysis in computer vision.
- Exploratory data analysis (EDA).

Not Appropriate For:

- Production deployment in sensitive domains (healthcare, security).
- Applications requiring balanced demographic representation without additional preprocessing.

Representativeness

- **Gender:** Strong skew toward masculine (33,240) and feminine (10,245) presentations, with very few non-binary (95).
- **Skin tone:** Heavier representation of lighter tones (`skin_tone_1–3` total ~20k+) compared to darker tones (`skin_tone_8–10` < 500 combined).
- **Age:** Most entries default to “not annotated,” limiting coverage.
- **Other categories:** Some professions/roles are uneven (e.g., “speaker,” “doctor,” “lawman” dominate).
- **Limitations:** Underrepresentation of marginalized groups makes the dataset less representative for fairness benchmarks; annotations are inferred rather than self-reported.

Data Quality

- **Missing data:** ~45,874 missing values, primarily in optional fields (hair type, lighting).
- **Duplicates:** None detected.
- **Noise:** Bounding boxes stored as JSON strings, not standardized tabular format.
- **Potential Biases:**
  - "NA" categories may bias models if treated as meaningful.
  - Skew in demographics (esp. non-binary and darker skin tones) risks spurious correlations.
- **Validation:** No clear evidence of external validation; labels appear to be manually or crowd-sourced, which can introduce subjectivity.

Pre-processing, Cleaning, Labeling

- Binary encodings used for most attributes (0/1).
- Gender is effectively treated as categorical but functionally binary in practice.
- Missing values retained as "NA" rather than imputed.
- Bounding boxes not yet parsed into separate numerical fields.

Privacy

- Dataset includes **sensitive demographic information** (gender presentation, skin tone, age).
- Risk of indirect re-identification if combined with other datasets.
- No explicit consent or provenance documentation included (important limitation).