FEHR & PEERS

MEMORANDUM

Date: March 1, 2019

To: Franz Loewenherz, City of Bellevue

From: Chris Breiland, Dana Weissman, Sarah Saviskas, and David Wasserman, Fehr & Peers

Subject: Task 3A – Value Added Research Findings

SE18-0634

Introduction

To complement the City of Bellevue's robust collision data analysis, we have leveraged some of Fehr & Peers' internal research and development funding to take a take a deeper dive into the City's 2010-2017collision data. The purpose of this value-added research is to better understand potential contributing/correlating factors with traffic collisions that may have applicability in other communities. By narrowing factors that are potentially related to traffic collisions, we can help Bellevue and other cities be more proactive and targeted in getting to the vision of zero serious injuries and fatalities in the future. The balance of this memorandum describes the analyses performed on the collision data. Overall, our analysis focused on the relationships between collisions and several key factors:

Geographic

- Top corridors for collisions
- High injury network

Land Use

- Adjacent land use designation
- Population and employment density
- Adjacency to schools

Speed and Volume

- Speed normalized by volumes

Equity and Demographics

- Low income and minority populations

Turning Vehicles

- Collisions by mode and vehicle turning movements

Geographic Relationships

Through our research of other Vision Zero cities, geographic analyses are often used to identify corridors or portions of the roadway network that have a disproportionate share of killed or seriously injured (KSI) collisions. A Vision Zero best practice is to identify a "High Injury Network" (HIN) that is a specific subset of the roadway network that can easily be mapped, and multiple City departments can prioritize for proactive education, enforcement, engineering, and engagement for the benefits of all modes. With this background in mind, we identified the top 10 corridors for all collisions, bicycle collisions, and pedestrian collisions, respectively.[1] These corridors are listed in Tables 1-3 below. The corridors were ranked by developing a total collision score.[2] Keep in mind that these corridors are not constrained by the HIN, but rather the HIN shows where the preponderance of collisions occur on these corridors. Of particular significance is where a corridor shows up in multiple tables, as these are corridors with high overall collision totals and high totals for pedestrians and/or bicycles.

When developing the collision scores, top ten corridors, and the HIN, KSI collisions are weighted more heavily than less-severe collisions. Our research indicates that there is no commonly accepted weight for KSI collisions, but in developing HINs and geographic collision analysis for nearly a dozen other communities, we typically apply a weight of 20 for KSI collisions. In other words, a single KSI collision is the equivalent of 20 non-severe collisions.

[1] Note that we focus on pedestrian and bicycle collisions because Bellevue data indicate that these modes are particularly vulnerable for KSI collisions. Bicycle and pedestrian modes are involved in about 5 percent of all collisions in Bellevue over the past 10 years, but they are involved in about 43 percent of all KSI collisions.

[2] The total collision score is a normalized index of the number of total collisions on each street, weighted by collision severity.

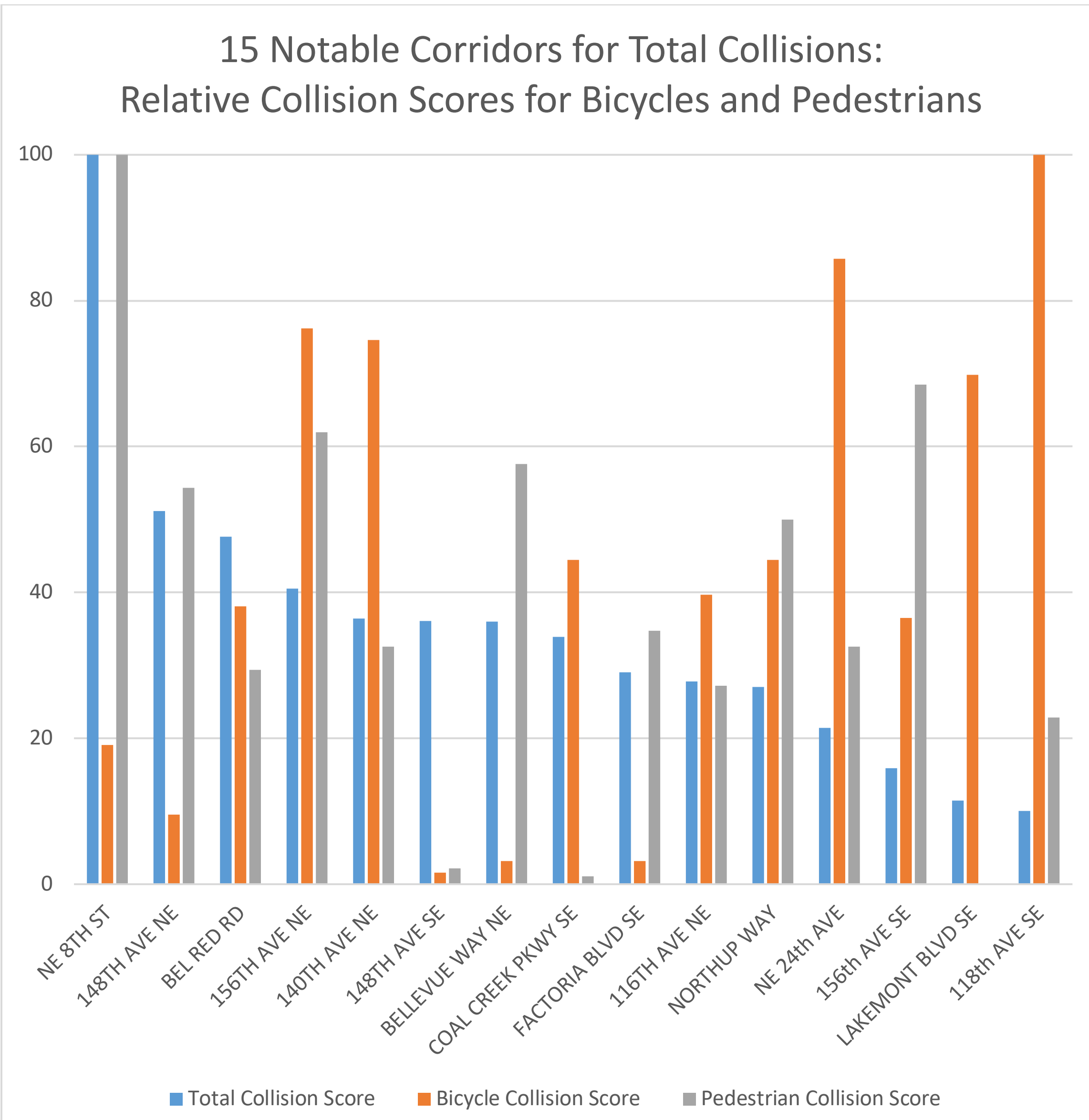
Table 1 – Top Ten Corridors for All Collisions	
Corridor	**Total Collision Score***
NE 8TH ST	100
148TH AVE NE	51
BEL RED RD	48
156TH AVE NE	41
140TH AVE NE	36
148TH AVE SE	36
BELLEVUE WAY NE	36
COAL CREEK PKWY SE	34
FACTORIA BLVD SE	29
116TH AVE NE	28

* This table was queried by identifying street name in the collision report. In this table, there are nearly twice as many collisions on NE 8th Street (weighted) as compared to 148th Ave NE.

Table 2 – Top Ten Corridors for Bicycle Collisions	
Corridor	**Total Collision Score**
118TH AVE SE	100
156TH AVE NE	76
140TH AVE NE	75
NE 24TH ST	71
LAKEMONT BLVD SE	70
NORTHUP WAY	44
COAL CREEK PKWY SE	44
116TH AVE NE	40
BEL RED RD	38
W LAKE SAMMAMISH PKWY SE	37

Table 3 – Top Ten Corridors for Pedestrian Collisions	
Corridor	**Total Collision Score**
NE 8TH ST	100
156TH AVE SE	68
156TH AVE NE	62
BELLEVUE WAY NE	58
MAIN ST	57
148TH AVE NE	54
NE 2ND ST	53
NE 4TH ST	53
NE 10TH ST	51
NORTHUP WAY	50

Tables 1-3 indicate that some corridors are present in the top 10 categories for each of the modes. For example, NE 8th Street is the top corridor for both total collisions and pedestrian collisions, but not for bicycle collisions. This fact is not surprising when considering that NE 8th Street is one of the longer and busier corridors in Bellevue and that it has a few nodes with major pedestrian activity – Downtown, Wilburton, Crossroads. At the same time, NE 8th Street is so busy that it is not an attractive bicycling route. The chart below highlights 15 major corridors and their relative collision scores for total, pedestrian, and bicycle modes.



Note in the chart above, the somewhat inverted pattern between total collision and bicycle collision scores. This may have something to do with the fact that busier auto-oriented streets are less attractive to bicyclists. A handful of streets, including 156th Avenue NE (near Crossroads), 140th Ave nue NE, and BelRed Road have relatively high collision scores when evaluating total, bicycle, and pedestrian collisions. Anecdotally, observations of activities in these areas indicate a relatively high proportion of pedestrians and bicycles, along with heavy vehicle traffic. Perhaps drivers are less attuned to active modes in these areas compared to other areas, particularly bicycles when compared to other activity centers in Bellevue.

To help put the corridor scores in more of a geographic context, we also developed a HIN. The HIN was created using the same weighting for KSI collisions as described earlier and is not specific to any particular mode. The HIN is shown in Figure 1 on the following page.

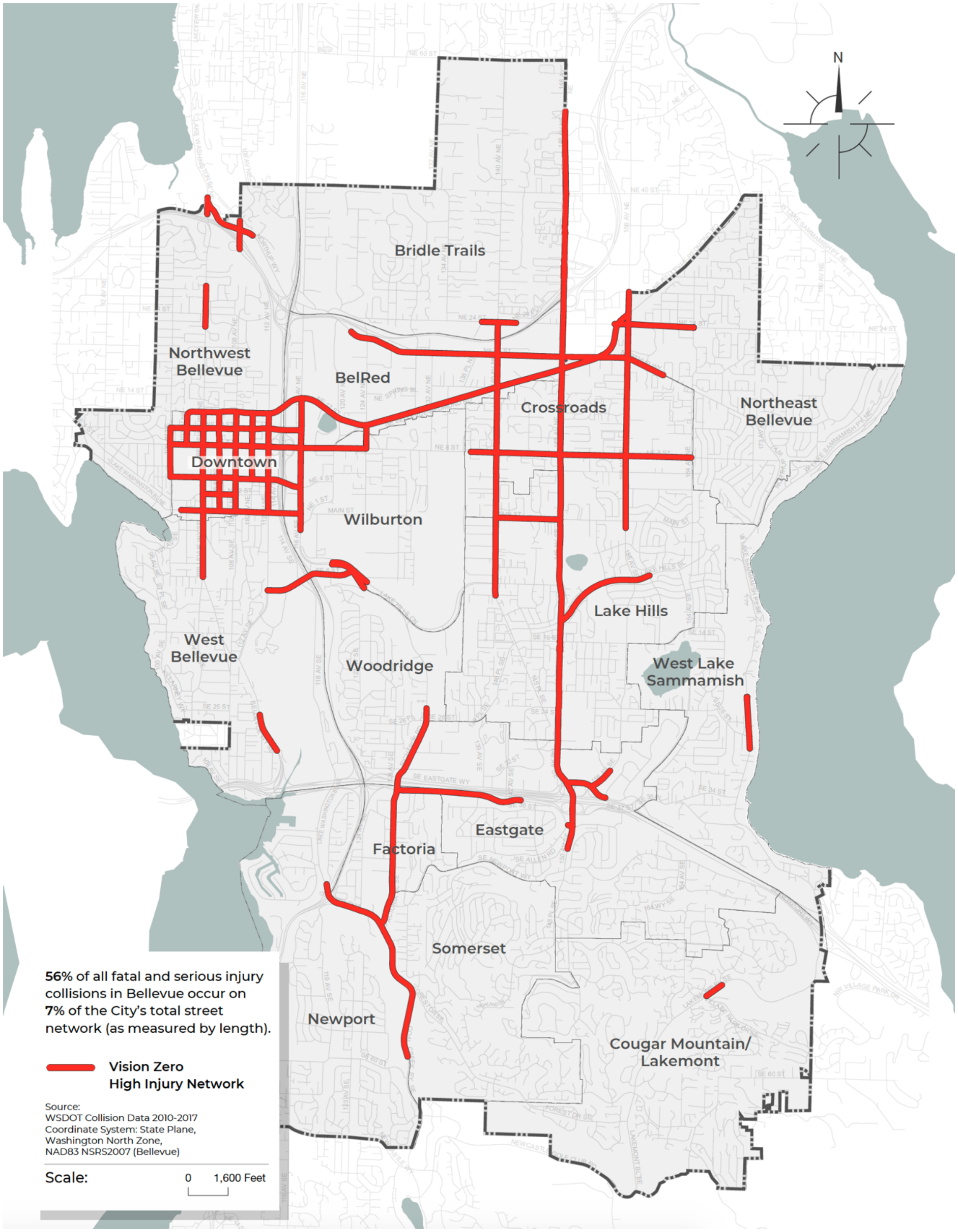


Figure 1 – Bellevue High Injury Network

As noted in the Figure 1 legend, the HIN covers just 7 percent of the City's street network, but includes 57 percent of all the KSI collisions. It is notable that busy areas like Downtown, Factoria, Overlake, and Crossroads are within the HIN – a pattern that we will see through other lenses throughout this document. While it is not necessarily surprising that the densest and busiest parts of the City have more collisions, the HIN does help concentrate where Bellevue should be investing in reducing KSI collisions. Notably, the busier corridors north of I-90 have a large number of the KSI collisions in Bellevue, and the Downtown network in general has many collisions.

Land Use Relationships

Our research of other Vision Zero cities suggests that certain land use types might have a greater number of collisions than others. However, few cities have done a systematic analysis of these relationships. To help better understand the relationships between land uses and traffic collisions, we prepared correlations related to adjacent land use designations from the Comprehensive Plan, population and employment density, and school proximity.

Land Use Designation

We compared collision data to the adjacent land use designations as identified in the Comprehensive Plan. Table 4 summarizes the results.

Table 4 – Collisions and Land Use

Land Use	Total Collisions	KSI Collisions	Acres	Percent of Acres	Percent or KSI Collisions	Percent of Total Collisions
Industrial	103	1	221	1%	1%	1%
Medical	162	1	136	1%	1%	1%
Mixed-Use	3,845	36	1,200	6%	23%	29%
Multi-Family	1,517	22	1,729	8%	14%	11%
Office	1,880	22	1,320	6%	14%	14%
Retail	2,004	14	579	3%	9%	15%
Single-Family	3,715	60	16,333	76%	38%	28%
Total	13,226	156	21,526	100%	100%	100%

A few notable results stand out from Table 4:

- 23 percent of all KSI collisions and 29 percent of all total collisions occur in Mixed Use land use area, which covers 6 percent of the City.
- 38 percent of all KSI collisions and 28 percent of all total collisions occur in Single-Family Residential areas, which covers 76 percent of the City.

Population and Employment Density

While Table 4 shows that a disproportionate share of KSI collisions occur in Mixed-Use areas of the City (which include areas like Downtown, Bel-Red, and Eastgate), these areas are also quite dense and there are more people who have the potential to be in a collision. To control for land use density, we also evaluated the number of total and KSI collisions by population and employment quintile (20 percent breaks in the population and employment). These density data come from the US EPA's Smart Location Database, which is a nationwide resource that maps more than 90 land use attributes including housing and employment density (https://www.epa.gov/smartgrowth/smart-location-mapping#SLD). If collisions and land use density were equally related, we would expect to see that about 20 percent of all collisions and KSI collisions would occur within each population and employment quintile. Tables 5 and 6 show the results of the analysis.

Table 5 – Collisions and Population Density

Population Density Quintile	Total Collisions	KSI Collisions	Acres	Percent of Acres	Percent or KSI Collisions	Percent of Total Collisions
0-20 (Low density)	4,422	50	7,426	31%	33%	32%
20-40	1,614	27	6,987	29%	12%	17%
40-60	1,145	18	4,270	18%	9%	12%
60-80	2,484	28	3,249	13%	19%	18%
80-100 (high density)	3,561	33	2,277	9%	27%	21%
Total	13,226	156	24,210	100%	100%	100%

Table 6 – Collisions and Employment Density

Population Density Quintile	Total Collisions	KSI Collisions	Acres	Percent of Acres	Percent or KSI Collisions	Percent of Total Collisions
0-20 (Low density)	981	21	7,703	32%	7%	13%
20-40	828	13	6,807	28%	6%	8%
40-60	1,774	20	4,423	18%	13%	13%
60-80	2,106	28	3,142	13%	16%	18%
80-100 (high density)	7,537	74	2,134	9%	57%	47%
Total	13,226	156	24,210	100%	100%	100%

Some notable findings from the population and density analysis include:

- The densest quintile for employment (which represents about 9 percent of the City) has about 47 percent of all KSI and 57 percent of total collisions in Bellevue; in other words, areas with high employment density have a disproportionately high incidence of KSI and total collisions, even when controlling for employment density.
- The distribution of KSI and total collisions are more evenly distributed across the population density quintiles with no strong patterns across high and low population density areas.

The results of the employment density analysis echo some findings from the HIN and land use designation analysis. Notably, areas with high employment densities, mixed-use and office zoning, and areas like Downtown, Overlake, portions of Crossroads, and Eastgate, have a disproportionately high rate of total and KSI collisions.

School Proximity

One other land use variable that we evaluated was the relationships between collisions and schools. Through this analysis, we reviewed all collisions within a quarter-mile of K-12 schools in Bellevue. Some key results are summarized in Table 7 below.

Table 7 – Collisions and Schools

Collision Location	Total Collisions	KSI Collisions (All Modes)	KSI Bicycle Collisions	KSI Pedestrian Collisions
Elsewhere	10,785	127	19	44
Near Schools	2,441	29	5	8
Total	13,226	156	24	52
Proportion of Collisions Near Schools	18%	19%	21%	15%

As can be seen in the table, the proportion of collisions near schools is fairly consistent at about 18-21 percent, which aligns with the proportion of land within a quarter-mile of schools. However, there is a lower proportion of KSI pedestrian collisions near schools, indicating that the pedestrian collisions that do occur near schools tend to be less severe. This may be related to the lower speeds near schools and more awareness of pedestrian activity at these locations.

Speed and Volume Relationships

It is well documented (and simple physics) that higher speeds result in more KSI collisions for people of all modes, but particularly for pedestrians and bicycles who are not protected within a vehicle. As identified by earlier City of Bellevue analysis, higher-speed streets have a disproportionate share of total and KSI collisions when compared to their total length.[3] Figure 2 highlights this relationship.

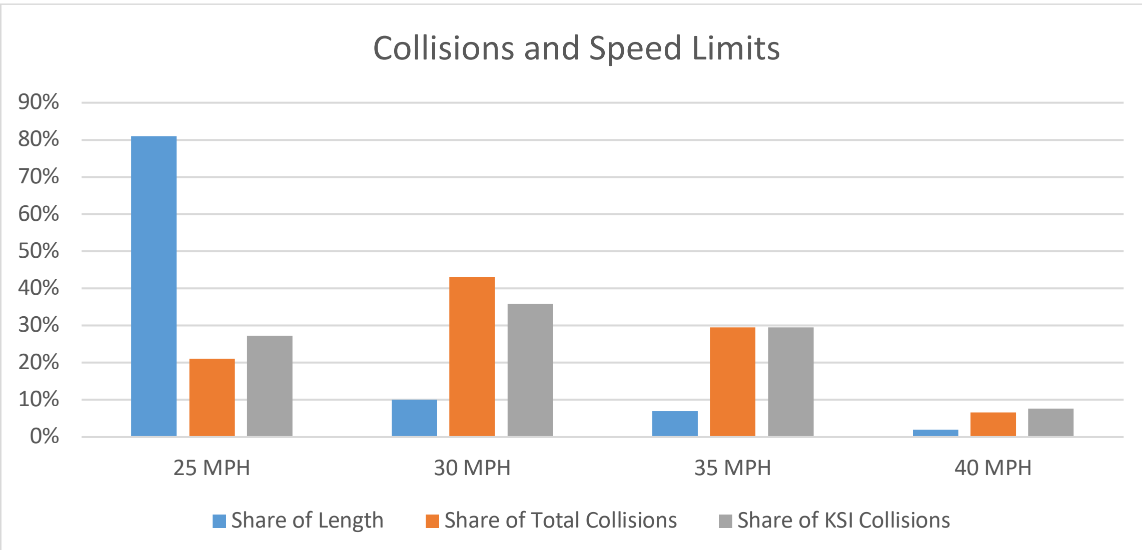


Figure 2 – Share of Roadway Speeds and Total/KSI Collisions

While the analysis summarized in Figure 2 provides us with some valuable data about speed limits and collisions, it does not control for how many vehicles are on the streets. In other words, a typical street with a speed limit of 25 MPH tends to have much less traffic than a typical street with a speed limit of 30 MPH (which includes the majority of arterials in Bellevue). Therefore, to control for traffic volumes, we used a combination of data from the City's traffic count database and "representative" traffic counts for roadway segments that Bellevue does not have observed data for. The representative data is the average observed ADT by functional classification with some manual adjustment if there is an adjacent count that can be used to refine the estimate. Using the observed and representative data, we estimate total vehicle-miles of travel (VMT) for each speed category

[3] Note that the analysis only includes City of Bellevue streets – WSDOT highways (I-405, I-90, SR 520) are not included in this analysis.

identified above. By estimating VMT, we are able to calculate a collision rate for each speed category, which controls for how much traffic is on the road. Table 8 summarizes the total and KSI collisions per million VMT over our analysis period.

Table 8 – Collisions, Speed, and VMT

Speed Limit	Total Collisions per Million VMT	KSI Collisions per Million VMT	Share of Total Length	Share of Total VMT	Share of KSI Collisions
25 MPH	5.986	0.0998	81%	37%	27%
30 MPH	7.255	0.0627	10%	30%	36%
35 MPH	5.618	0.0743	7%	25%	29%
40 MPH	6.287	0.1149	2%	8%	8%

Table 8 shows that total collision rates are higher for streets with 30 MPH speeds. The KSI collision rate is highest for the roads with speeds of 40 MPH or greater (which represent a relatively small proportion of Bellevue's overall road network), which is generally expected. However, *the second highest KSI collision rate is on streets with a speed limit or 25 MPH, which is not typical for other cities*. A deeper dive into the data shows that the KSI rate for pedestrians and bicycles may be influencing the overall rates, since these modes have relatively high rates, even at these lower speeds. One factor to note in this analysis is that we have less observed data for streets with 25 MPH, and our approach of using representative data might overstate the VMT on these roadways. Additional data would help to refine this approach. Table 9 shows the KSI rate per VMT for bicycles and pedestrians.

Table 9 – Bicycle and Pedestrian KSI Rates by Speed

Speed Limit	Bicycle KSI Collisions per Million VMT	Pedestrian KSI Collisions per Million VMT
25 MPH	0.033	0.033
30 MPH	0.010	0.028
35 MPH	0.006	0.025
40 MPH	0.011	0.034

As shown in Table 9, the KSI rate for bicycles in particular is high for the 25 MPH streets (and also relatively high for pedestrians). This result is likely due to the higher rates of people walking and biking along slower (and generally lower vehicle volume) streets. Therefore, the collision rate per VMT likely overstates the risk to people walking and biking along 25 MPH streets, but we do not have consistent citywide data on bicycle or pedestrian volumes. Regardless, these findings highlight what other City data already point out—the risk of being killed or seriously injured is much higher for bicycles or pedestrians involved in a collision compared to people in a vehicle. Minimizing the risk to vulnerable bicyclists and pedestrians should be an important goal of Bellevue's Vision Zero strategy.

Equity and Demographics

Bellevue City staff provided Census block groups in Bellevue with a high prevalence of low income populations and people of color. Using these data, we reviewed the collision data to determine if these areas have a higher prevalence of total and KSI collisions. The low income and high minority population areas constitute about 15 percent of the City's land area and 19 percent of the population. Figure 3 on the following page shows a map of these areas.

The results of the equity and demographic analysis indicates the following:

- 27 percent of all collisions
- 35 percent of all pedestrian collisions
- 25 percent of all bicycle collisions
- 23 percent of all KSI collisions

As can be inferred from the bullets above, when accounting for land area and population, areas of low income and high minority populations are slightly over-represented for total, KSI, and bicycle collisions. However, there is a more pronounced disparity related to pedestrian collisions in these areas.

Turning Vehicles

Recent analysis by New York City (https://bit.ly/2pvoPa5) has shed light on the significance of pedestrian and bicycle safety and turning vehicles. Based on these insights, we took a closer examination of how turning vehicles are related to pedestrian and bicycle safety.

Perhaps the most striking outcome of our turning vehicle data review is the fact that bicycle collisions are coded differently in the collision reports than other collision types. Notably, the direction of travel (turning, going straight, etc.) for vehicle-bicycle collisions is rarely recorded, while it is commonly recorded for pedestrian and vehicle-vehicle collisions. Therefore, we were unable to derive any insights about the significance of turning vehicles and bicycle safety. However, anecdotal information from Bellevue's recent (January 2019) public survey on traffic safety highlighted a few notable stories/examples of bicyclists that were hit by vehicles making turns.

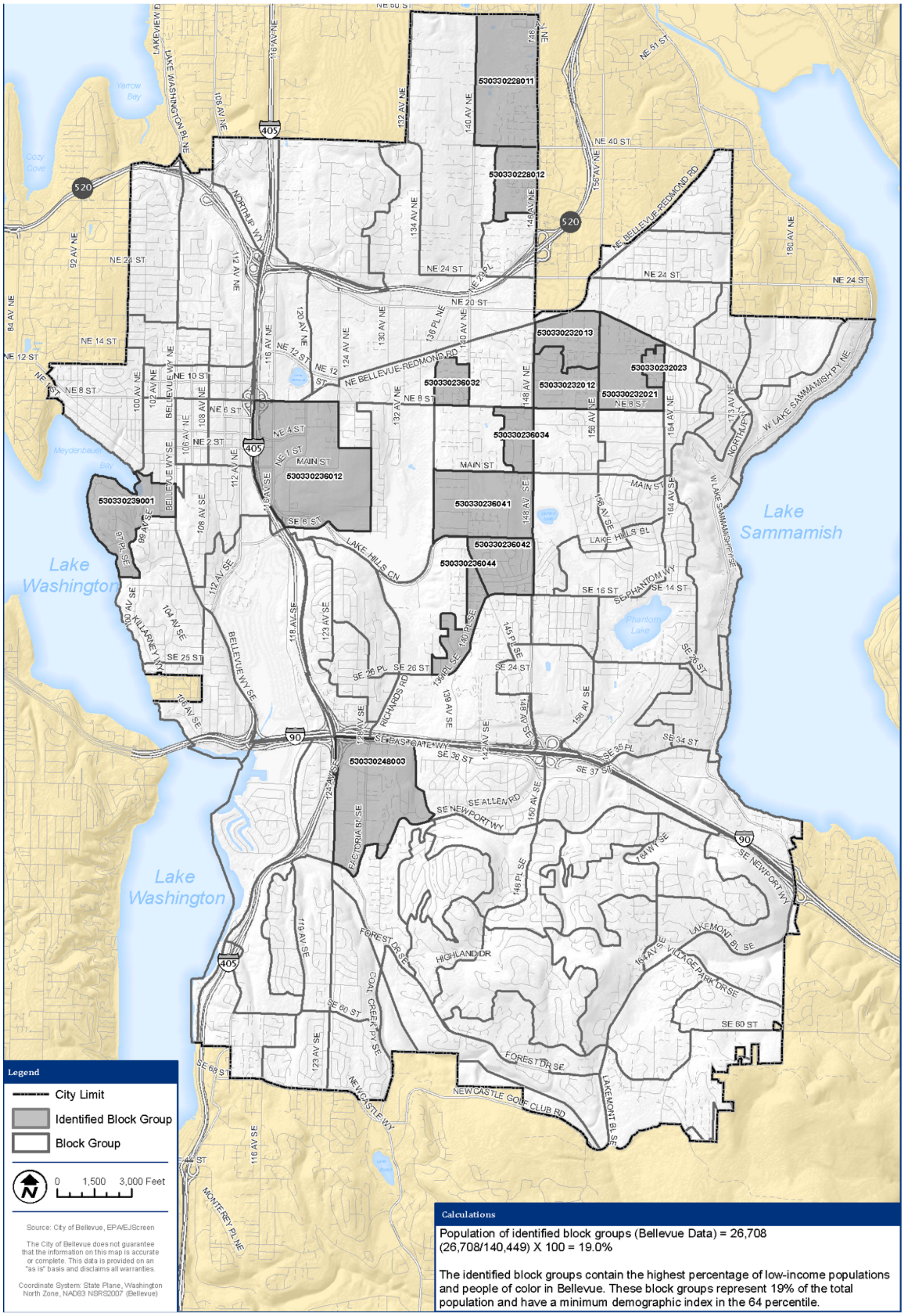


Figure 3 – Areas of Low Income and High Minority Populations

Below are some notable findings of the pedestrian and turning vehicle data analysis:

- About two-thirds of all pedestrian collisions are related to turning vehicles; this compares to 20 percent for vehicle-vehicle collisions.
- For pedestrians, collisions involving left-turns are 1.4 times more likely to lead to a KSI outcome compared to right turns; pedestrian collisions with a vehicle traveling straight are 2.4 times more likely to result in a KSI collision compared to right turns. This is likely due to the speed of the vehicle (right-turning vehicles tend to travel slower than left-turn or through vehicles).
- 35 percent of all pedestrian collisions are related to right turning vehicles; this compares to 29 percent for left turning vehicles. There is room to better educate vehicles to watch for pedestrians when turning.

In addition to turning vehicle collision analysis, we worked to evaluate collisions relative to traffic signal operations (cycle length, left-turn phasing, etc.). However, given the complexities of Bellevue's advanced adaptive traffic signal system, we did not have enough time/data to perform a detailed analysis and gain any insight from traffic signal operations. However, data from other cities indicates that protected left-turn phasing tends to have safer outcomes for pedestrians, bicycles, and vehicles. Future analysis of traffic signal operations and safety outcomes could be beneficial to help the City balance mobility and safety outcomes.

Conclusions

The value-added research yielded a number of notable relationships from a geographic, land use, speed/volume, equity/demographics, and turning vehicles perspective. Below are some key take-aways:

- A large proportion of KSI collisions occur on a relatively small length of total City of Bellevue streets. A High Injury Network (HIN) was identified that covers about 7 percent of the City's streets, but includes 56 percent of all the KSI collisions.
- A disproportionate share of total and KSI collisions occur in the City's mixed-use areas, particularly those areas with high employment density. This higher share of collisions is disproportionate in terms of both the total land area covered by mixed-use areas and activity-levels generated by the dense employment areas. While we expect higher-density areas to have higher total numbers of total and KSI collisions, the magnitude of the difference in collision totals (particularly for high employment density areas) was larger than we expected. As Bellevue continues to densify, this trend warrants monitoring.
- Areas around schools have a "typical" level of total and KSI collisions, although the severity of pedestrian collisions tends to be lower, potentially due to lower speeds and more driver awareness.
- Collision rates for total and KSI collisions generally tend to increase with speed, which is expected. However, the KSI collision rate for low-speed, 25 MPH roads, was the second

highest in the City. This factor may have to do with the fact that pedestrians and bicyclists use these lower-volume roads at a greater rate, which skews the KSI rate per VMT up.

- Areas of the City with higher proportions of low income and minority populations have a somewhat higher rate of total and KSI collisions when considering population and geographic extent. However, the rate of pedestrian collisions was notably higher in these areas.
- Turning vehicles are involved with a disproportionate number of pedestrian collisions, when compared to vehicle-vehicle collisions. While anecdotal evidence hints that this trend might also be true for bicycles, there is no data to substantiate this hypothesis because of limitations of how bicycle collision data is recorded.

While the analysis results above span a range of topics, they can all help to focus Bellevue's Vision Zero program to achieve the greatest outcome. In short, the higher-speed streets in the denser, mixed-use areas of the City should be areas of attention for Bellevue. Additionally, helping to reduce vehicle turning collisions on these streets could go far to help protect the more vulnerable users of the transportation system—pedestrians and bicyclists.

Appendix A

This appendix contains the raw output from Fehr & Peers GIS queries on the Bellevue collision data.

Bellevue_Vision_Zero_Analysis

This notebook documents Fehr & Peers metric development efforts for Bellevue's Vision Zero Action Plan. Guidelines are as follows:

Metric development should occur using the shared library or copies of it. If you need to move it locally, merge your code additions to the shared library.

Develop metrics in 1-2 cells per metric by using function calls to the library with functions you add.

If different files are created, the format assumed for an easy merge is.

Markdown Cell: Metric Name

Python Cell: Operations in python not more than 15 lines. If longer, add abstractions in sharedlib. Plots/Data QAQC should be done in separate cells to be removed later.

In [1]:
```
import pandas as pd
import arcpy
import numpy as np
import seaborn as sns
import os
import CollisionProfileLib as CP
from IPython.display import HTML
arcpy.env.overwriteOutput = True
```

In [2]:
```
pd.set_option('display.max_rows', 100)
pd.set_option('display.max_columns', 100)
```

Set Up Paths.

In [3]:
```
main_dir = r"N:\Projects\Non_SanJose_Projects\SE Projects\SE18-0634_Be
llevue_Vision_Zero\Analysis_DW\Model"
project_db = os.path.join(main_dir,"Project_Data.gdb")
base_fds = os.path.join(project_db,"Base_Data")
street_network = os.path.join(base_fds,"Final_HIN_VS4_BellevueSt")
net_copy = os.path.join(base_fds,"Correlation_Network")
collisions = os.path.join(base_fds,"AllCollisions_CityStreets_LX_DataW
SDOT_2010_2017_Clipped")
final_joined_collisions = os.path.join(base_fds,"AllCollisions_CityStr
eets_LX_DataWSDOT_2010_2017_WSpatialDataJned")
in_mem = "in_memory"
```

Associate Collision Data With Other Datasets

Associate collision data to various contextual datasets using Near and Spatial Joins.

In [4]:
```
temp_collisions = os.path.join(in_mem,"CollisionsCurrent")
temp_table = os.path.join(in_mem,"TemporaryTable")
temp_fc = os.path.join(in_mem,"TemporaryFC")
arcpy.CopyFeatures_management(collisions,temp_collisions)
arcpy.CopyFeatures_management(temp_collisions,final_joined_collisions)
print("Temporary and Final Collision File Created.")
```

```
Temporary and Final Collision File Created.
```

Get Collision Types

This section has cross tabulations of the following collision variables.:

a. Raw Counts |b. KSI| c. Bike | d. Pedestrian | e. Non-BikePed (vehicle to vehicle)|f. Lighting Conditions| g. First Collision Movement Classification | h. Road Surface Condition (Wet/Dry/Etc) | i. DUI (taken from driver fields)

Top 10 Corridors by Weighted Collisions (All, Bike, Ped)

In [5]:
```
original_fields = ['PRIMARY_TR','ROADWAY_SU', 'LIGHTING_C', 'FIRST_COL
L', 'MV_DRIVER_', 'MV_DRIVER1', 'MV_DRIVE_1', 'MV_DRIVE_2', 'MV_DRIVE_
3', 'MV_DRIVE_4','KSI', 'KSI_Bike', 'KSI_Ped', 'KSI_NoBP', 'Wtd_ColAll
', 'Wtd_ColBic', 'Wtd_ColPed', 'Wtd_NoBkPed']
col_df = CP.arcgis_table_to_df(temp_collisions,original_fields)
pivot = pd.pivot_table(col_df,index = ["PRIMARY_TR"], values=['Wtd_Col
All', 'Wtd_ColBic', 'Wtd_ColPed', 'Wtd_NoBkPed'],aggfunc="sum")
print("Top 10 Corridors by All KSI Weighted Collisions")
pivot.sort_values("Wtd_ColAll",ascending=False).head(10).style.bar()
```

Top 10 Corridors by All KSI Weighted Collisions

Out[5]:

	Wtd_ColAll	Wtd_ColBic	Wtd_ColPed	Wtd_NoBkPed
PRIMARY_TR				
NE 8TH ST	1388	12	92	1284
148TH AVE NE	710	6	50	654
BEL RED RD	661	24	27	610
156TH AVE NE	563	48	57	458
140TH AVE NE	506	47	30	429
148TH AVE SE	501	1	2	498
BELLEVUE WAY NE	500	2	53	445
COAL CREEK PKWY SE	471	28	1	442
FACTORIA BLVD SE	403	2	32	369
116TH AVE NE	386	25	25	336

In [6]:

```
print("Top 10 Corridors by Bicycle KSI Weighted Collisions")
pivot.sort_values("Wtd_ColBic",ascending=False).head(10).style.bar()
```

Top 10 Corridors by Bicycle KSI Weighted Collisions

Out[6]:

	Wtd_ColAll	Wtd_ColBic	Wtd_ColPed	Wtd_NoBkPed
PRIMARY_TR				
118TH AVE SE	139	63	21	55
156TH AVE NE	563	48	57	458
140TH AVE NE	506	47	30	429
NE 24TH ST	298	45	30	223
LAKEMONT BLVD SE	159	44	0	115
NORTHUP WAY	375	28	46	301
COAL CREEK PKWY SE	471	28	1	442
116TH AVE NE	386	25	25	336
BEL RED RD	661	24	27	610
W LAKE SAMMAMISH PKWY SE	162	23	0	139

In [7]:
```
print("Top 10 Corridors by Pedestrian KSI Weighted Collisions")
pivot.sort_values("Wtd_ColPed",ascending=False).head(10).style.bar()
```

```
Top 10 Corridors by Pedestrian KSI Weighted Collisions
```

Out[7]:

	Wtd_ColAll	Wtd_ColBic	Wtd_ColPed	Wtd_NoBkPed
PRIMARY_TR				
NE 8TH ST	1388	12	92	1284
156TH AVE SE	221	23	63	135
156TH AVE NE	563	48	57	458
BELLEVUE WAY NE	500	2	53	445
MAIN ST	366	1	52	313
148TH AVE NE	710	6	50	654
NE 2ND ST	126	1	49	76
NE 4TH ST	354	0	49	305
NE 10TH ST	231	1	47	183
NORTHUP WAY	375	28	46	301

In [8]:

```
weighted_cols = [i for i in col_df.columns if "Wtd_" in i]
for weight_col in weighted_cols:
    col_df[weight_col.replace("Wtd_","")] = np.where(col_df[weight_col
]>0,1,0)
all_mode_columns =  ["ColAll","ColBic","ColPed","NoBkPed"]
all_ksi_columns = [i for i in col_df.columns if "KSI" in i]
all_mode_ksi_cols = all_mode_columns + all_ksi_columns
print("KSI Collisions by Mode")
pd.pivot_table(col_df,index = ["KSI"], values=all_mode_columns,aggfunc
="sum",margins=True,margins_name="Total")
```

KSI Collisions by Mode

Out[8]:

	ColAll	ColBic	ColPed	NoBkPed
KSI				
0.0	13070	203	333	12534
1.0	156	24	52	80
Total	13226	227	385	12614

In [9]:

```
mv_drive_cols = [i for i in col_df.columns if "MV_DRIVE" in i] # Add I
ntoxicated Pedestrians (Focus)
for ind,drive_col in enumerate(mv_drive_cols):
    if ind==0:
        col_df["Combined_Drive"] = col_df[drive_col]
    else:
        col_df["Combined_Drive"] = col_df["Combined_Drive"] + ":" +col
_df[drive_col]
col_df["Drive_DUI"] = np.where(col_df["Combined_Drive"].str.contains("
Under Influence"),1,0)
print("Collisions by Mode With a Driver DUI Involved")
pd.pivot_table(col_df,index = ["Drive_DUI"], values=all_mode_columns,a
ggfunc="sum",margins=True,margins_name="Total")
```

Collisions by Mode With a Driver DUI Involved

Out[9]:

	ColAll	ColBic	ColPed	NoBkPed
Drive_DUI				
0	12790	226	383	12181
1	436	1	2	433
Total	13226	227	385	12614

In [10]:
```
directional_fields = col_df
dir_df = pd.pivot_table(directional_fields,index = ["FIRST_COLL"], val
ues=["ColAll","ColBic","ColPed","NoBkPed","KSI"],aggfunc="sum",margins
=True,margins_name="Total")
cm = sns.light_palette("red", as_cmap=True,n_colors=20)
dir_df.style.background_gradient(cmap=cm)
```

Out[10]:

	ColAll	ColBic	ColPed	KSI	NoBkPed
FIRST_COLL					
All other non-collision	6	0	0	0	6
Boulder (stationary)	15	0	0	1	15
Bridge Column, Pier or Pillar	2	0	0	0	2
Bridge Rail - Face	7	0	0	2	7
Building	16	0	1	0	15
Concrete Barrier/Jersey Barrier - Face	10	0	0	0	10
Concrete Barrier/Jersey Barrier - Leading End	1	0	0	0	1
Culvert and/or other Appurtenance in Ditch	6	0	0	0	6
Curb, Raised Traffic Island or Raised Median Curb	38	0	0	2	38
Domestic animal (horse, cow, sheep, etc)	1	0	0	0	1
Earth Bank or Ledge	11	0	0	2	11
Entering at angle	2853	0	5	15	2848
Fence	72	0	0	1	72
Fire Hydrant	19	0	0	0	19
From opposite direction - all others	122	0	0	0	122
From opposite direction - both going straight - one stopped - sideswipe	9	0	0	0	9
From opposite direction - both going straight - sideswipe	35	0	0	2	35
From opposite direction - both moving - head-on	28	0	0	1	28
From opposite direction - one left turn - one right turn	120	0	0	0	120
From opposite direction - one left turn - one					

straight	1501	2	2	20	1497
From opposite direction - one stopped - head-on	12	0	0	0	12
From same direction - all others	293	0	0	0	293
From same direction - both going straight - both moving - rear-end	707	0	0	1	707
From same direction - both going straight - both moving - sideswipe	1232	0	1	2	1231
From same direction - both going straight - one stopped - rear-end	3500	1	0	6	3499
From same direction - both going straight - one stopped - sideswipe	87	0	0	0	87
From same direction - one left turn - one straight	108	0	0	0	108
From same direction - one right turn - one straight	254	1	0	0	253
Guardrail - Face	40	0	1	1	39
Guardrail - Leading End	2	0	0	0	2
Guardrail - Through, Over or Under	2	0	0	0	2
Guide Post	1	0	0	0	1
Linear Curb	31	0	0	1	31
Mailbox	27	0	0	1	27
Manhole Cover	6	0	0	0	6
Metal Sign Post	71	0	0	0	71
Miscellaneous Object or Debris on Road	4	0	0	0	4
Not Stated	4	0	0	0	4
Not stated	1	0	0	0	1
One car entering parked position	10	0	0	0	10
One car leaving parked position	46	0	0	0	46
One parked--one moving	327	0	2	4	325
Other Objects	47	0	0	0	47
Over Embankment - No Guardrail Present	11	0	0	0	11
Person fell, jumped or was pushed from	1	0	0	1	1

vehicle					
Retaining Wall (concrete, rock, brick, etc.)	53	0	0	1	53
Roadway Ditch	29	0	0	4	29
Rock Bank or Ledge	5	0	0	0	5
Same direction -- both turning left -- both moving -- rear end	8	0	0	0	8
Same direction -- both turning left -- both moving -- sideswipe	77	0	0	0	77
Same direction -- both turning left -- one stopped -- rear end	11	0	0	0	11
Same direction -- both turning right -- both moving -- rear end	16	0	0	0	16
Same direction -- both turning right -- both moving -- sideswipe	51	0	0	0	51
Same direction -- both turning right -- one stopped -- rear end	116	0	0	0	116
Same direction -- both turning right -- one stopped -- sideswipe	4	0	0	0	4
Signal Pole	54	0	0	2	54
Street Light Pole or Base	90	0	0	1	90
Temporary Traffic Sign, Barricade or Construction Materials	5	0	0	0	5
Traffic Island	9	0	0	0	9
Tree or Stump (stationary)	202	0	0	7	202
Underside of Bridge	1	0	0	0	1
Utility Box	13	0	0	1	13
Utility Pole	56	0	0	2	56
Vehicle - Pedalcyclist	223	223	0	23	0
Vehicle Strikes Deer	2	0	0	0	2
Vehicle backing hits pedestrian	5	0	5	1	0
Vehicle going straight hits pedestrian	117	0	117	22	0
Vehicle hits Pedestrian - All Other Actions	5	0	5	3	0
Vehicle overturned	67	0	0	2	67

Vehicle turning left hits pedestrian	110	0	110	12	0
Vehicle turning right hits pedestrian	135	0	135	11	0
Wood Sign Post	66	0	1	1	65
Total	13226	227	385	156	12614

Compute Colliisions by Movement

Look to see if a right, left, or straight movement is involved in any collision. This is done by searching all the options above for right,left, and straight in the "FIRST_COLL" field. Keep in mind that there is double counting in this tabulation (a collision involve a right and left turn for exmaple).

In [11]:

```
col_df["LEFT_TURN_INVOLVED"] =  np.where(col_df["FIRST_COLL"].str.contains("left"),1,0)
col_df["RIGHT_TURN_INVOLVED"] = np.where(col_df["FIRST_COLL"].str.contains("right"),1,0)
col_df["MOVING_STRAIGHT_INVOLVED"] = np.where(col_df["FIRST_COLL"].str.contains("straight"),1,0)
pd.pivot_table(directional_fields,index = ["LEFT_TURN_INVOLVED"], values=all_mode_ksi_cols,aggfunc="sum",margins=True,margins_name="Total").reindex(all_mode_ksi_cols,axis=1)
```

Out[11]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	KSI_Ped	K
LEFT_TURN_INVOLVED								
0	11291	225	273	10793	124.0	23.0	39.0	6
1	1935	2	112	1821	32.0	1.0	13.0	1
Total	13226	227	385	12614	156.0	24.0	52.0	8

In [12]:

```
pd.pivot_table(directional_fields,index = ["RIGHT_TURN_INVOLVED"], values=all_mode_ksi_cols,aggfunc="sum",margins=True,margins_name="Total").reindex(all_mode_ksi_cols,axis=1)
```

Out[12]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	KSI_Ped
RIGHT_TURN_INVOLVED							
0	12530	226	250	12054	145.0	24.0	41.0
1	696	1	135	560	11.0	0.0	11.0
Total	13226	227	385	12614	156.0	24.0	52.0

In [13]:
```
pd.pivot_table(directional_fields,index = ["MOVING_STRAIGHT_INVOLVED"]
, values=all_mode_ksi_cols,aggfunc="sum",margins=True,margins_name="To
tal").reindex(all_mode_ksi_cols,axis=1)
```

Out[13]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	KS
MOVING_STRAIGHT_INVOLVED							
0	5676	223	265	5188	103.0	23.0	29
1	7550	4	120	7426	53.0	1.0	23
Total	13226	227	385	12614	156.0	24.0	52

Collisions by Surface Conditions

In [14]:
```
print("All vs. KSI Collisions by Surface Conditions")
pd.pivot_table(col_df,index = ['ROADWAY_SU'], values=all_mode_ksi_cols
,aggfunc="sum",margins=True,margins_name="Total").reindex(all_mode_ksi
_cols,axis=1)
```

```
All vs. KSI Collisions by Surface Conditions
```

Out[14]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	KSI_Ped	KSI_NoBP
ROADWAY_SU								
	3	0	0	3	0.0	0.0	0.0	0.0
Dry	9027	192	251	8584	107.0	18.0	33.0	56.0
Ice	53	1	3	49	2.0	1.0	0.0	1.0
Other	11	0	0	11	1.0	0.0	0.0	1.0
Sand/Mud/Dirt	5	0	0	5	0.0	0.0	0.0	0.0
Snow/Slush	39	0	1	38	2.0	0.0	1.0	1.0
Standing Water	15	0	0	15	0.0	0.0	0.0	0.0
Unknown	88	0	3	85	0.0	0.0	0.0	0.0
Wet	3985	34	127	3824	44.0	5.0	18.0	21.0
Total	13226	227	385	12614	156.0	24.0	52.0	80.0

Collisions by Lighting Condition

In [15]:

```
print("All vs. KSI Collisions by Lighting Condition")
pd.pivot_table(col_df,index = ["LIGHTING_C"], values=["ColAll","KSI"],
aggfunc="sum",margins=True,margins_name="Total")
```

```
All vs. KSI Collisions by Lighting Condition
```

Out[15]:

	ColAll	KSI
LIGHTING_C		
	2	0.0
Dark-No Street Lights	163	2.0
Dark-Street Lights Off	37	0.0
Dark-Street Lights On	2899	57.0
Dawn	138	1.0
Daylight	9558	92.0
Dusk	352	3.0
Other	3	1.0
Unknown	74	0.0
Total	13226	156.0

Proximity To Schools

This section evaluates how being within 1/4 mile of different school types relates to collision counts. ES represents elementary schools, MS is middle schools, HS is Highschools, and Coll is college. KSI collisions are tabulated by quarter mile band, then all collisions by mode/KSI are reported by whether or not they are near an Elementary,Middle, or Highschool (Primary to Secondary Education).

In [16]:
```python
schools = os.path.join(base_fds,"Bellevue_Schools")
temp_school = os.path.join(in_mem,"SchoolTemp")
school_fields = []
for value in ["ES","MS","HS","COLL"]:
    print("Processing value ", value)
    arcpy.Select_analysis(schools,temp_school, where_clause="TYPE = '{
0}'".format(value))
    field_name = "{0}_Sch_Qrt_Mi".format(value)
    field_dist = field_name+"_DIST"
    arcpy.GenerateNearTable_analysis(temp_collisions, temp_school,temp
_table, search_radius="0.5 Miles", closest="ALL")
    sch_df = CP.arcgis_table_to_df(temp_table,["IN_FID","NEAR_FID","NE
AR_DIST"])
    sch_df[field_name] = np.where(sch_df["NEAR_DIST"] < 1320,1, 0)
    sch_df[field_dist] = sch_df["NEAR_DIST"]
    grp_by_df = sch_df.groupby(by="IN_FID")
    summary_df = grp_by_df.agg({field_dist:"min",field_name:"max"})
    summary_df = summary_df[[field_name,field_dist]]
    col_df = pd.merge(col_df,summary_df,left_index=True,right_index=Tr
ue,how = 'left')
    col_df = col_df.fillna({field_name:0,field_dist:-1})
    school_fields.append(field_name)
col_df["PrimaryToSecondary_Sch_Qrt_Mi"] = (col_df["ES_Sch_Qrt_Mi"] + c
ol_df["HS_Sch_Qrt_Mi"] + col_df["MS_Sch_Qrt_Mi"]).clip(0,1)
school_fields.append("PrimaryToSecondary_Sch_Qrt_Mi")
# Add School Summary to HIN Tabulations
print("KSI Collisions by Whether They Are 1/4 Mile Away from a School"
)
pd.pivot_table(col_df,index = ["KSI"], values= school_fields,aggfunc="
sum",margins=True,margins_name="Total")
```

```
Processing value  ES
Processing value  MS
Processing value  HS
Processing value  COLL
KSI Collisions by Whether They Are 1/4 Mile Away from a School
```

Out[16]:

	COLL_Sch_Qrt_Mi	ES_Sch_Qrt_Mi	HS_Sch_Qrt_Mi	MS_Sch_Qrt_Mi	PrimaryToS
KSI					
0.0	438.0	1672.0	613.0	573.0	2412.0
1.0	5.0	23.0	5.0	5.0	29.0
Total	443.0	1695.0	618.0	578.0	2441.0

In [17]:
```
print("Number of Collisions by Mode & KSI that are within 1/4 Mile of an Elementary, Middle, or High School")
pd.pivot_table(col_df,index = ["PrimaryToSecondary_Sch_Qrt_Mi"], values= all_mode_ksi_cols,aggfunc="sum",margins=True,margins_name="Total").reindex(all_mode_ksi_cols,axis=1)
```

```
Number of Collisions by Mode & KSI that are within 1/4 Mile of an Elementary, Middle, or High School
```

Out[17]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	K
PrimaryToSecondary_Sch_Qrt_Mi							
0.0	10785	182	315	10288	127.0	19.0	4
1.0	2441	45	70	2326	29.0	5.0	8
Total	13226	227	385	12614	156.0	24.0	5

Adjacent Land Use

In [18]:

```
join_fc = os.path.join(base_fds,"Bellevue_Comprehensive_Plan")
join_fields = ["GeneralLUC","ComplanDes"]
f_map = CP.generate_statistical_fieldmap(temp_collisions,join_fc,merge
_rule_dict={"FIRST":join_fields})
arcpy.SpatialJoin_analysis(temp_collisions,join_fc,temp_fc,match_optio
n="CLOSEST",field_mapping=f_map)
#IF the list has a string inside of it, one of the fields has a partia
l match to the
new_fields = [i.name for i in arcpy.ListFields(temp_fc) if ["Field Mat
ch" for j in join_fields if j in i.name]]
summary_df = CP.arcgis_table_to_df(temp_fc,new_fields)
summary_df.columns = join_fields
col_df = pd.merge(col_df,summary_df,left_index=True,right_index=True,h
ow = 'left')
pd.pivot_table(col_df,index = join_fields[0], values=all_mode_ksi_cols
,aggfunc="sum",margins=True,margins_name="Total").reindex(all_mode_ksi
_cols,axis=1)
```

Out[18]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	KSI_Ped	KSI_NoBP
GeneralLUC								
Light Industrial	103	2	1	100	1.0	0.0	1.0	0.0
Medical	162	1	6	155	1.0	0.0	0.0	1.0
Mixed-Use	3845	39	124	3682	36.0	3.0	21.0	12.0
Multi-family	1517	29	50	1438	22.0	2.0	8.0	12.0
Office	1880	35	41	1804	22.0	4.0	6.0	12.0
Retail	2004	22	69	1913	14.0	1.0	4.0	9.0
Single-family	3715	99	94	3522	60.0	14.0	12.0	34.0
Total	13226	227	385	12614	156.0	24.0	52.0	80.0

Land Use Area Calculations

In [19]:
```
lu_fc = os.path.join(base_fds,"Bellevue_Comprehensive_Plan")
lu_fields =  ["GeneralLUC","ComplanDes","SHAPE@"]
lu_df = CP.arcgis_table_to_df(lu_fc,lu_fields)
lu_df["ACRES"] = lu_df["SHAPE@"].apply(lambda x: x.getArea(units="ACRE
S"))
lu_pivot = pd.pivot_table(lu_df,index = lu_fields[0], values=["ACRES"]
,aggfunc="sum",margins=True,margins_name="Total")
lu_pivot["Percent of Area"] = lu_pivot["ACRES"]/lu_pivot.loc["Total","
ACRES"] * 100
lu_pivot
```

Out[19]:

	ACRES	Percent of Area
GeneralLUC		
Camp and Conference Center	9.316584	0.043280
Light Industrial	220.662499	1.025086
Medical	135.549679	0.629695
Mixed-Use	1200.164661	5.575356
Multi-family	1729.005733	8.032083
Office	1319.631238	6.130337
Retail	578.745644	2.688558
Single-family	16333.167927	75.875605
Total	21526.243964	100.000000

Bike Facilities

(Only Evaluates Right Side Existing Facilities- check on codes later)

In [20]:
```
join_fc = os.path.join(base_fds,"Bellevue_Bike_Network")
join_fields = ["RIGHTEXIST","LEFTEXIST"]
search_radius = "100 Feet"
f_map = CP.generate_statistical_fieldmap(temp_collisions,join_fc,merge
_rule_dict={"FIRST":join_fields})
arcpy.SpatialJoin_analysis(temp_collisions,join_fc,temp_fc,match_optio
n="CLOSEST",field_mapping=f_map,search_radius=search_radius )
#IF the list has a string inside of it, one of the fields has a partia
l match to the
new_fields = [i.name for i in arcpy.ListFields(temp_fc) if ["Field Mat
ch" for j in join_fields if j in i.name]]
summary_df = CP.arcgis_table_to_df(temp_fc,new_fields)
summary_df.columns = join_fields
col_df = pd.merge(col_df,summary_df,left_index=True,right_index=True,h
ow = 'left')
pd.pivot_table(col_df,index = join_fields[0], values=all_mode_ksi_cols
,aggfunc="sum",margins=True,margins_name="Total").reindex(all_mode_ksi
_cols,axis=1)
# Talk to Bianca - correlate it - Off-St
```

Out[20]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	KSI_Ped	KSI_NoBP
RIGHTEXIST								
	5754	79	154	5521	54.0	4.0	18.0	32.0
B	2674	77	66	2531	45.0	13.0	14.0	18.0
C	877	12	27	838	11.0	0.0	3.0	8.0
D	709	22	25	662	12.0	2.0	5.0	5.0
E	357	5	15	337	3.0	0.0	1.0	2.0
F	173	5	6	162	3.0	1.0	0.0	2.0
G	56	3	0	53	2.0	1.0	0.0	1.0
OFFST	51	0	0	51	0.0	0.0	0.0	0.0
Total	10651	203	293	10155	130.0	21.0	41.0	68.0

Priority Development Areas

In [21]:

```
join_fc = os.path.join(base_fds,"Bellevue_Priority_Census_Block_Groups
")
join_fields = ["Priority_Census_Block_Groups"]
search_radius = "10 Feet"
f_map = CP.generate_statistical_fieldmap(temp_collisions,join_fc,merge
_rule_dict={"FIRST":join_fields})
arcpy.SpatialJoin_analysis(temp_collisions,join_fc,temp_fc,match_optio
n="CLOSEST",field_mapping=f_map,search_radius=search_radius )
#IF the list has a string inside of it, one of the fields has a partia
l match to the
new_fields = [i.name for i in arcpy.ListFields(temp_fc) if ["Field Mat
ch" for j in join_fields if j in i.name]]
summary_df = CP.arcgis_table_to_df(temp_fc,new_fields)
summary_df.columns = join_fields
col_df = pd.merge(col_df,summary_df,left_index=True,right_index=True,h
ow = 'left')
col_df = col_df.fillna({i:0 for i in join_fields})
pd.pivot_table(col_df,index = join_fields[0], values=all_mode_ksi_cols
,aggfunc="sum",margins=True,margins_name="Total").reindex(all_mode_ksi
_cols,axis=1)
```

Out[21]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	KSI
Priority_Census_Block_Groups							
0.0	9528	169	250	9109	120.0	22.0	39.(
1.0	3698	58	135	3505	36.0	2.0	13.(
Total	13226	227	385	12614	156.0	24.0	52.(

Priority Area Summary Statistics from EJ Screen

After associating the priority census tracts with EJ screen data, service population, low income population, minority population, and coverages are calculated by priority area.

In [22]:
```
ej_screen = os.path.join(base_fds, "Bellevue_Only_EJScreen_WPriorityId
entified")
ej_fields =  ["Priority_Census_Tracts", "ACSTOTPOP","LOWINCOME","MINOR
POP","LESSHS","AREALAND","SHAPE@"]
ej_df = CP.arcgis_table_to_df(ej_screen,ej_fields)
ej_df["ACRES"] = lu_df["SHAPE@"].apply(lambda x: x.getArea(units="ACRE
S"))
ej_pivot = pd.pivot_table(ej_df,index = ej_fields[0], values=["ACSTOTP
OP","LOWINCOME","MINORPOP","LESSHS","AREALAND","ACRES"] ,aggfunc="sum"
,margins=True,margins_name="Total")
ej_pivot["Percent of Area"] = ej_pivot["ACRES"]/ej_pivot.loc["Total","
ACRES"] * 100
ej_pivot
```

Out[22]:

	ACRES	ACSTOTPOP	AREALAND	LESSHS	LOWINCOM
Priority_Census_Tracts					
0	2636.970383	104274	76386718.0	2624	14333
1	462.965001	28727	9897150.0	1647	8188
Total	3099.935384	133001	86283868.0	4271	22521

HIN/Street Characteristics

This section has more than just data associations, but multiple cross tabulations of different street characteristics such as HIN, Speed Limits, Freight Routes, and other characteristics.

In [23]:
```
join_fc = os.path.join(base_fds,"Final_HIN_VS7_BellevueSt")
join_fields = ["SpeedLimit","Oneway","FunctionCl","TruckRoute","HIN_75
_Clean","DN_Wtd_ColAll","DN_Wtd_ColPed","DN_Wtd_ColBic","DN_Wtd_NoBkPe
d"]
search_radius = "25 Feet"
f_map = CP.generate_statistical_fieldmap(temp_collisions,join_fc,merge
_rule_dict={"FIRST":join_fields})
arcpy.SpatialJoin_analysis(temp_collisions,join_fc,temp_fc,match_optio
n="CLOSEST",field_mapping=f_map,search_radius=search_radius )
#IF the list has a string inside of it, one of the fields has a partia
l match to the
new_fields = [i.name for i in arcpy.ListFields(temp_fc) if ["Field Mat
ch" for j in join_fields if j in i.name]]
summary_df = CP.arcgis_table_to_df(temp_fc,new_fields)
print(new_fields)
print(join_fields)
summary_df.columns = join_fields
col_df = pd.merge(col_df,summary_df,left_index=True,right_index=True,h
ow = 'left')
col_df = col_df.fillna({i:0 for i in join_fields})
col_df.describe()
```

```
['FIRSTSpeedLimit', 'FIRSTOneway', 'FIRSTFunctionCl', 'FIRSTTruckRou
te', 'FIRSTHIN_75_Clean', 'FIRSTDN_Wtd_ColAll', 'FIRSTDN_Wtd_ColPed'
, 'FIRSTDN_Wtd_ColBic', 'FIRSTDN_Wtd_NoBkPed']
['SpeedLimit', 'Oneway', 'FunctionCl', 'TruckRoute', 'HIN_75_Clean',
'DN_Wtd_ColAll', 'DN_Wtd_ColPed', 'DN_Wtd_ColBic', 'DN_Wtd_NoBkPed']
```

Out[23]:

	KSI	KSI_Bike	KSI_Ped	KSI_NoBP	Wtd_ColAll	Wtd_
count	13226.000000	13226.000000	13226.000000	13226.000000	13226.000000	13226.
mean	0.011795	0.001815	0.003932	0.006049	1.224104	0.0516
std	0.107966	0.042561	0.062582	0.077541	2.051359	0.8594
min	0.000000	0.000000	0.000000	0.000000	1.000000	0.0000
25%	0.000000	0.000000	0.000000	0.000000	1.000000	0.0000
50%	0.000000	0.000000	0.000000	0.000000	1.000000	0.0000
75%	0.000000	0.000000	0.000000	0.000000	1.000000	0.0000
max	1.000000	1.000000	1.000000	1.000000	20.000000	20.000

In [24]:
```
print("Make 20 MPH the Minimum Speed Limit")
col_df["SpeedLimit"] = np.where(col_df["SpeedLimit"]<=20,20,col_df["Sp
eedLimit"])
col_df["SpeedLimit"].unique()
```

```
Make 20 MPH the Minimum Speed Limit
```

Out[24]:
```
array([40., 25., 30., 35., 20.])
```

In [25]:

```
print("All Mode and KSI Collisions by Speed Limit")
pd.pivot_table(col_df,index =["SpeedLimit"], values=all_mode_ksi_cols,
aggfunc="sum",margins=True,margins_name="Total").reindex(all_mode_ksi_
cols,axis=1)
```

```
All Mode and KSI Collisions by Speed Limit
```

Out[25]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	KSI_Ped	KSI_NoBP
SpeedLimit								
20.0	219	5	15	199	4.0	1.0	2.0	1.0
25.0	2518	49	101	2368	38.0	5.0	10.0	23.0
30.0	5712	110	169	5433	56.0	12.0	22.0	22.0
35.0	3911	49	94	3768	46.0	5.0	15.0	26.0
40.0	866	14	6	846	12.0	1.0	3.0	8.0
Total	13226	227	385	12614	156.0	24.0	52.0	80.0

In [26]:

```
print("All Mode and KSI Collisions by Location on High Injury Network"
)
pd.pivot_table(col_df,index =["HIN_75_Clean"], values=all_mode_ksi_col
s ,aggfunc="sum",margins=True,margins_name="Total").reindex(all_mode_k
si_cols,axis=1)
```

```
All Mode and KSI Collisions by Location on High Injury Network
```

Out[26]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	KSI_Ped	KSI_NoBP
HIN_75_Clean								
0.0	5382	121	160	5101	69.0	13.0	19.0	37.0
1.0	7844	106	225	7513	87.0	11.0	33.0	43.0
Total	13226	227	385	12614	156.0	24.0	52.0	80.0

Summarize Network Length by Facility Type

In [27]:
```
join_fc = os.path.join(base_fds,"Final_HIN_VS7_BellevueSt")
join_fields = ["SpeedLimit","Oneway","FunctionCl","TruckRoute","HIN_75
_Clean","SHAPE@"]
street_df = CP.arcgis_table_to_df(join_fc,join_fields)
street_df["SpeedLimit"] = np.where(street_df["SpeedLimit"]<=20,20,stre
et_df["SpeedLimit"])
street_df["MILES"] = street_df["SHAPE@"].apply(lambda x: x.getLength(u
nits="MILES"))
pd.pivot_table(street_df,index =["SpeedLimit"],columns =["HIN_75_Clean
"], values=["MILES"],aggfunc="sum",margins=True,margins_name="Total")
```

Out[27]:

	MILES		
HIN_75_Clean	**0.0**	**1.0**	**Total**
SpeedLimit			
20	38.884955	NaN	38.884955
25	386.367300	1.214886	387.582186
30	31.822165	19.390490	51.212655
35	23.280754	14.536621	37.817375
40	8.053422	3.446437	11.499859
Total	488.408596	38.588434	526.997031

Collision Rates and AADT Normalization

The collision rate is assumed to be calculated as:

Collision Rate

$$\frac{(Accidents * 1000,000)}{(AnnualVMT)}$$

This tranlates to:

$$\frac{(Accidents * 1000,000)}{(WeekdayAADT * 350 * 7 * MilesofNetwork)}$$

We use 350 rather than 365 because weekday VMT is likely higher than weekend VMT. To give some account for this we use a smaller multiplier. The 7 is to account for the fact we have 7 years of collision data.

In [72]:

```
# Ammount AADT Roads
join_fc = os.path.join(base_fds,"Final_HIN_VS7_BellevueSt_WAADT")
join_fields = ["SpeedLimit","FunctionCl","AADT_Final","HIN_75_Clean","
SHAPE@"]
aadt_df = CP.arcgis_table_to_df(join_fc,join_fields).fillna(0)
aadt_df["SpeedLimit"] = np.where(aadt_df["SpeedLimit"]<=20,20,aadt_df[
"SpeedLimit"])
aadt_df["MILES"] = aadt_df["SHAPE@"].apply(lambda x: x.getLength(units
="MILES"))
aadt_df["AADT_MI_Product"] = aadt_df["AADT_Final"] * aadt_df["MILES"]
aadt_pivot = pd.pivot_table(aadt_df,index =["SpeedLimit"], values=["MI
LES","AADT_MI_Product"], aggfunc="sum",margins=True, margins_name="Tot
al")
aadt_pivot["Weighted Avg AADT"] = aadt_pivot["AADT_MI_Product"]/ aadt_
pivot["MILES"]
aadt_pivot["VMT Per Day"] =  aadt_pivot["Weighted Avg AADT"] * aadt_pi
vot["MILES"] # Same as AADT_MI_Product
aadt_pivot["VMT Per Year"] =  aadt_pivot["VMT Per Day"] * 350
aadt_pivot["VMT Over Study Period"] =  aadt_pivot["VMT Per Year"] * 7
speed_pivot = pd.pivot_table(col_df,index =["SpeedLimit"], values=all_
mode_ksi_cols,aggfunc="sum",margins=True,margins_name="Total").reindex
(all_mode_ksi_cols,axis=1)
collision_aadt_miles = pd.merge(aadt_pivot,speed_pivot,left_index=True
,right_index=True)
collision_aadt_miles
```

Out[72]:

	AADT_MI_Product	MILES	Weighted Avg AADT	VMT Per Day	VMT Per Year
SpeedLimit					
20	4.124211e+04	38.884955	1060.618713	4.124211e+04	1.443474e+07
25	6.067063e+05	387.582186	1565.361615	6.067063e+05	2.123472e+08
30	5.174126e+05	51.212655	10103.216970	5.174126e+05	1.810944e+08
35	4.258439e+05	37.817375	11260.536484	4.258439e+05	1.490454e+08
40	1.444548e+05	11.499859	12561.440738	1.444548e+05	5.055918e+07
Total	1.735660e+06	526.997031	3293.490447	1.735660e+06	6.074809e+08

In [73]:
```
col_ksi_columns = [i for i in collision_aadt_miles if "Col" in i or "K
SI" in i or "NoBkP" in i]
collision_per_mile = collision_aadt_miles.copy()
for i in col_ksi_columns:
    collision_per_mile[i+"_Per_1Mil_VMT"]  = ((collision_per_mile[i]*1
000000.0)/collision_per_mile["VMT Over Study Period"])
    collision_per_mile = collision_per_mile.drop(i,axis=1)
print("Collision Rates Across Network by Speed Limit")
collision_per_mile.style.background_gradient(cmap=cm)
```

```
Collision Rates Across Network by Speed Limit
```

Out[73]:

	AADT_MI_Product	MILES	Weighted Avg AADT	VMT Per Day	VMT Per Year	VMT Ov Study Period
SpeedLimit						
20	41242.1	38.885	1060.62	41242.1	1.44347e+07	1.01043
25	606706	387.582	1565.36	606706	2.12347e+08	1.48643
30	517413	51.2127	10103.2	517413	1.81094e+08	1.26766
35	425844	37.8174	11260.5	425844	1.49045e+08	1.04332
40	144455	11.4999	12561.4	144455	5.05592e+07	3.53914
Total	1.73566e+06	526.997	3293.49	1.73566e+06	6.07481e+08	4.25237

In [74]:
```
hin_filtered_aadt_df = aadt_df[aadt_df["HIN_75_Clean"]==1].copy()
hin_aadt_pivot = pd.pivot_table(hin_filtered_aadt_df,index =["SpeedLim
it"], values=["MILES","AADT_MI_Product"], aggfunc="sum",margins=True,
margins_name="Total")
hin_aadt_pivot["Weighted Avg AADT"] = hin_aadt_pivot["AADT_MI_Product"
]/ hin_aadt_pivot["MILES"]
hin_aadt_pivot["VMT Per Day"] =  hin_aadt_pivot["Weighted Avg AADT"] *
hin_aadt_pivot["MILES"] # Same as AADT_MI_Product
hin_aadt_pivot["VMT Per Year"] =  hin_aadt_pivot["VMT Per Day"] * 350
hin_aadt_pivot["VMT Over Study Period"] =  hin_aadt_pivot["VMT Per Yea
r"] * 7
hin_only_col_df = col_df[col_df["HIN_75_Clean"]==1].copy()
speed_pivot = pd.pivot_table(hin_only_col_df,index =["SpeedLimit"], va
lues=all_mode_ksi_cols,aggfunc="sum",margins=True,margins_name="Total"
).reindex(all_mode_ksi_cols,axis=1)
collision_aadt_miles = pd.merge(hin_aadt_pivot,speed_pivot,left_index=
True,right_index=True)
col_ksi_columns = [i for i in collision_aadt_miles if "Col" in i or "K
SI" in i or "NoBkP" in i]
collision_per_mile = collision_aadt_miles.copy()
for i in col_ksi_columns:
    collision_per_mile[i+"_Per_1Mil_VMT"]  = ((collision_per_mile[i]*1
000000.0)/collision_per_mile["VMT Over Study Period"])
    collision_per_mile = collision_per_mile.drop(i,axis=1)
print("Collision Rates on HIN Network by Speed Limit")
collision_per_mile.style.background_gradient(cmap=cm)
```

Collision Rates on HIN Network by Speed Limit

Out[74]:

	AADT_MI_Product	MILES	Weighted Avg AADT	VMT Per Day	VMT Per Year	VMT Over Study Period
SpeedLimit						
25	12273.1	1.21489	10102.3	12273.1	4.29559e+06	3.00691e+07
30	247306	19.3905	12754	247306	8.65571e+07	6.059e+08
35	197693	14.5366	13599.7	197693	6.91926e+07	4.84348e+08
40	35512.8	3.44644	10304.2	35512.8	1.24295e+07	8.70063e+07
Total	492785	38.5884	12770.3	492785	1.72475e+08	1.20732e+09

Smart Location Database Associations

In addition to spatial joins, this cell bins the analysis into 5 quintile bins for each census geography before the join. Column Identies:

- D1A- Housing Unit Density (Per Acre)
- D1B- Population Density (Per Acre)
- D1C -Job Density (Per Acre)
- D2A_JPHH - Job Housing Balance
- D3b- Intersection Density
- D4c - Transit Accessibility

See User Guide for Details: https://www.epa.gov/smartgrowth/smart-location-database-technical-documentation-and-user-guide (https://www.epa.gov/smartgrowth/smart-location-database-technical-documentation-and-user-guide)

In [30]:
```
join_fc = os.path.join(base_fds,"Bellevue_Only_SmartLocationDB")
join_fields = ["D1a","D1b","D1c","D2A_JPHH","D3b","D4c"]
CP.add_Percentile_Fields(join_fc,join_fields)
```

```
Creating percentile column for field D1a.
Creating percentile column for field D1b.
Creating percentile column for field D1c.
Creating percentile column for field D2A_JPHH.
Creating percentile column for field D3b.
Creating percentile column for field D4c.
Exporting new percentile dataframe to structured numpy array.
Joining new standarized fields to feature class. The new fields are
['Perc_D1a', 'Perc_D1b', 'Perc_D1c', 'Perc_D2A_JPHH', 'Perc_D3b', 'P
erc_D4c', 'DFIndexJoin']
Script Completed Successfully.
```

In [31]:

```
percentile_fields = ["Perc_D1a", "Perc_D1b", "Perc_D1c", "Perc_D2A_JPH
H", "Perc_D3b", "Perc_D4c"]
all_fields = join_fields + percentile_fields
search_radius = "25 Feet"
f_map = CP.generate_statistical_fieldmap(temp_collisions,join_fc,merge
_rule_dict={"FIRST":all_fields})
arcpy.SpatialJoin_analysis(temp_collisions,join_fc,temp_fc,match_optio
n="CLOSEST",field_mapping=f_map,search_radius=search_radius )
#IF the list has a string inside of it, one of the fields has a partia
l match to the
new_fields = [i.name for i in arcpy.ListFields(temp_fc) if ["Field Mat
ch" for j in join_fields if j in i.name]]
summary_df = CP.arcgis_table_to_df(temp_fc,new_fields)
summary_df.columns = all_fields
col_df = pd.merge(col_df,summary_df,left_index=True,right_index=True,h
ow = 'left')
col_df = col_df.fillna({i:0 for i in all_fields})
col_df[all_fields].describe()
# Add Area Covered by SLD quartiles
# Bin by Quartile
```

Out[31]:

	D1a	D1b	D1c	D2A_JPHH	D3b	
count	13226.000000	13226.000000	13226.000000	13226.000000	13226.000000	13226.
mean	5.282753	8.243688	20.544010	6.854027	79.718788	117.67
std	6.120945	6.673227	31.278584	8.919404	44.047218	75.135
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.0000
25%	1.929810	3.705833	1.522838	0.442308	49.215199	62.000
50%	2.873701	6.628177	7.431520	3.526906	73.064636	108.00
75%	5.529527	9.787690	20.992206	9.553750	99.672890	152.66
max	29.335618	35.540567	143.971200	33.630631	246.538910	363.66

In [32]:
```
quintile_columns = []
quintile_index = [-.01,.2,.4,.6,.8,1.1]
for i in percentile_fields:
    new_col = i.replace("Perc","Quintile_Category")
    col_df[new_col] = pd.cut(col_df[i],quintile_index,labels=["<20th Percentile","<40th Percentile","<60th Percentile","<80th Percentile","<100th Percentile"])
    quintile_columns.append(new_col)
col_df[quintile_columns].head()
```

Out[32]:

	Quintile_Category_D1a	Quintile_Category_D1b	Quintile_Category_D1c	Qu
OBJECTID				
1	<20th Percentile	<20th Percentile	<20th Percentile	<2
2	<60th Percentile	<80th Percentile	<40th Percentile	<4
3	<20th Percentile	<20th Percentile	<20th Percentile	<2
4	<60th Percentile	<80th Percentile	<40th Percentile	<4
5	<20th Percentile	<20th Percentile	<20th Percentile	<2

In [33]:
```
pd.pivot_table(col_df,index =["Quintile_Category_D1a"], values=all_mode_ksi_cols,aggfunc="sum",margins=True,margins_name="Total").reindex(all_mode_ksi_cols,axis=1).style.bar()
```

Out[33]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	KSI_Ped	KSI
Quintile_Category_D1a								
<20th Percentile	4422	79	94	4249	50	10	13	27
<40th Percentile	1614	49	38	1527	27	6	7	14
<60th Percentile	1145	21	33	1091	18	4	4	10
<80th Percentile	2484	33	79	2372	28	2	11	15
<100th Percentile	3561	45	141	3375	33	2	17	14
Total	13226	227	385	12614	156	24	52	80

In [34]:
```
pd.pivot_table(col_df,index =["Quintile_Category_D1b"], values=all_mode_ksi_cols,aggfunc="sum",margins=True,margins_name="Total").reindex(all_mode_ksi_cols,axis=1).style.bar()
```

Out[34]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	KSI_Ped	KSI
Quintile_Category_D1b								
<20th Percentile	4523	85	83	4355	51	10	10	31
<40th Percentile	1877	52	60	1765	29	7	10	12
<60th Percentile	1849	26	65	1758	23	4	9	10
<80th Percentile	2109	23	48	2038	22	1	5	16
<100th Percentile	2868	41	129	2698	31	2	18	11
Total	13226	227	385	12614	156	24	52	80

In [35]:
```
pd.pivot_table(col_df,index =["Quintile_Category_D1c"], values=all_mode_ksi_cols,aggfunc="sum",margins=True,margins_name="Total").reindex(all_mode_ksi_cols,axis=1).style.bar()
```

Out[35]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	KSI_Ped	KSI
Quintile_Category_D1c								
<20th Percentile	981	26	19	936	21	3	5	13
<40th Percentile	828	23	21	784	13	2	3	8
<60th Percentile	1774	33	49	1692	20	5	6	9
<80th Percentile	2106	31	58	2017	28	5	9	14
<100th Percentile	7537	114	238	7185	74	9	29	36
Total	13226	227	385	12614	156	24	52	80

Smart Location Database Area Tabulations

Quintiles summarzied by area of the city.

In [36]:

```
sld_fc = os.path.join(base_fds,"Bellevue_Only_SmartLocationDB")
sld_df = CP.arcgis_table_to_df(sld_fc,percentile_fields+["SHAPE@"])
for i in percentile_fields:
    new_col = i.replace("Perc","Quintile_Category")
    sld_df[new_col] = pd.cut(sld_df[i],quintile_index,labels=["<20th P
ercentile","<40th Percentile","<60th Percentile","<80th Percentile","<
100th Percentile"])
sld_df["ACRES"] = sld_df["SHAPE@"].apply(lambda x: x.getArea(units="AC
RES"))
sld_pivot = pd.pivot_table(sld_df,index = quintile_columns[0], values=
["ACRES"] ,aggfunc="sum",margins=True,margins_name="Total")
sld_pivot["Percent of Area"] = sld_pivot["ACRES"]/sld_pivot.loc["Total
","ACRES"] * 100
sld_pivot
```

Out[36]:

	ACRES	Percent of Area
Quintile_Category_D1a		
<20th Percentile	7426.267161	30.674889
<40th Percentile	6986.729555	28.859338
<60th Percentile	4269.780396	17.636726
<80th Percentile	3249.488269	13.422314
<100th Percentile	2277.332256	9.406733
Total	24209.597637	100.000000

In [37]:

```
sld_pivot = pd.pivot_table(sld_df,index = quintile_columns[1], values=
["ACRES"] ,aggfunc="sum",margins=True,margins_name="Total")
sld_pivot["Percent of Area"] = sld_pivot["ACRES"]/sld_pivot.loc["Total
","ACRES"] * 100
sld_pivot
```

Out[37]:

	ACRES	Percent of Area
Quintile_Category_D1b		
<20th Percentile	7702.629854	31.816431
<40th Percentile	6807.461257	28.118853
<60th Percentile	4423.462678	18.271525
<80th Percentile	3142.342167	12.979737
<100th Percentile	2133.701680	8.813454
Total	24209.597637	100.000000

In [38]:

```
sld_pivot = pd.pivot_table(sld_df,index = quintile_columns[2], values=
["ACRES"] ,aggfunc="sum",margins=True,margins_name="Total")
sld_pivot["Percent of Area"] = sld_pivot["ACRES"]/sld_pivot.loc["Total
","ACRES"] * 100
sld_pivot
```

Out[38]:

	ACRES	Percent of Area
Quintile_Category_D1c		
<20th Percentile	4887.582953	20.188617
<40th Percentile	5179.347417	21.393777
<60th Percentile	5141.643556	21.238038
<80th Percentile	3582.423322	14.797534
<100th Percentile	5418.600389	22.382034
Total	24209.597637	100.000000

Signal Proximity and Character Associations

This portion of the analysis is dedicated to comparing how proximity to signalized intersections by type relates with accident type and their characteristics.

In [40]:
```python
search_radius = "100 Feet"
f_map = CP.generate_statistical_fieldmap(temp_collisions,join_fc,merge_rule_dict={"MAX":join_fields})
arcpy.SpatialJoin_analysis(temp_collisions,join_fc,temp_fc,match_option="INTERSECT",field_mapping=f_map,search_radius=search_radius )
#IF the list has a string inside of it, one of the fields has a partial match to the
new_fields = [i.name for i in arcpy.ListFields(temp_fc) if ["Field Match" for j in join_fields if j in i.name]]
summary_df = CP.arcgis_table_to_df(temp_fc,new_fields)
join_fields = [i.replace("SynchroDataOSMMatch_","") for i in join_fields]
summary_df.columns = join_fields
col_df = pd.merge(col_df,summary_df,left_index=True,right_index=True,how = 'left')
col_df = col_df.fillna({i:0 for i in join_fields})
col_df[join_fields].describe()
```

Out[40]:

	Prot	Perm	ProtPerm	Signalized_Intersection
count	13226.000000	13226.000000	13226.000000	13226.000000
mean	0.132996	0.013458	0.204521	0.435581
std	0.339583	0.115231	0.403367	0.495852
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	0.000000	0.000000
50%	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000

In [41]:
```python
pd.pivot_table(col_df,index =["Signalized_Intersection"], values=all_mode_ksi_cols,aggfunc="sum",margins=True,margins_name="Total").reindex(all_mode_ksi_cols,axis=1)
```

Out[41]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	KSI_Ped	K
Signalized_Intersection								
0.0	7465	169	217	7079	107.0	23.0	34.0	5
1.0	5761	58	168	5535	49.0	1.0	18.0	3
Total	13226	227	385	12614	156.0	24.0	52.0	8

In [42]:

```
col_df["Signal Type"]  = np.where(col_df["Signalized_Intersection"]==1
, "Signalized Intersection With No Data","Non-Intersection")
col_df["Signal Type"]  = np.where(col_df["Perm"]==1, "Permitted",col_d
f["Signal Type"] )
col_df["Signal Type"]  = np.where(col_df["ProtPerm"]==1 , "Protected-P
ermitted",col_df["Signal Type"] )
col_df["Signal Type"]  = np.where(col_df["Prot"]==1 , "Protected",col_
df["Signal Type"])
pd.pivot_table(col_df,index =["Signal Type"], values=all_mode_ksi_cols
,aggfunc="sum",margins=True,margins_name="Total").reindex(all_mode_ksi
_cols,axis=1)
```

Out[42]:

	ColAll	ColBic	ColPed	NoBkPed	KSI	KSI_Bike	KSI_Ped	KSI_NoBP
Signal Type								
Non-Intersection	7465	169	217	7079	107.0	23.0	34.0	50.0
Permitted	134	3	9	122	1.0	0.0	1.0	0.0
Protected	1759	15	39	1705	7.0	0.0	3.0	4.0
Protected-Permitted	2705	27	74	2604	27.0	1.0	7.0	19.0
Signalized Intersection With No Data	1163	13	46	1104	14.0	0.0	7.0	7.0
Total	13226	227	385	12614	156.0	24.0	52.0	80.0

In [43]:
```
pd.pivot_table(col_df,index =["Signal Type"], values=["MOVING_STRAIGHT
_INVOLVED","RIGHT_TURN_INVOLVED","LEFT_TURN_INVOLVED"], aggfunc="sum",
margins=True,margins_name="Total")
```

Out[43]:

	LEFT_TURN_INVOLVED	MOVING_STRAIGHT_INVOLVED	RIGHT_TURN_II
Signal Type			
Non-Intersection	699	4009	233
Permitted	27	71	13
Protected	216	1110	146
Protected-Permitted	690	1725	194
Signalized Intersection With No Data	303	635	110
Total	1935	7550	696

Count of Signals by Type

This approach is not perfect because some of the ids were very proximal or overlapping. This however does provide a rough series of numbers of the number of signalized intersections.

In [44]:
```
join_fc = os.path.join(base_fds,"Bellevue_Traffic_Signals_OSM")
temp_buff = os.path.join(in_mem,"Sig_Buff")
temp_buffsp = os.path.join(in_mem,"Sig_BuffSP")
temp_join = os.path.join(in_mem,"Sig_Join")
merged_intersections = os.path.join(base_fds,"Bellevue_Traffic_Signals
_OSM_Merged")
join_fields = ["SynchroDataOSMMatch_Prot","SynchroDataOSMMatch_Perm","
SynchroDataOSMMatch_ProtPerm","Signalized_Intersection"]
arcpy.Buffer_analysis(join_fc,temp_buff, dissolve_option="ALL",buffer_
distance_or_field="100 Feet")
arcpy.MultipartToSinglepart_management(temp_buff,temp_buffsp)
field_map = CP.generate_statistical_fieldmap(temp_buffsp,join_fc,merge
_rule_dict={"MAX":join_fields})
arcpy.SpatialJoin_analysis(temp_buffsp,join_fc,temp_join,field_mapping
=field_map)
arcpy.FeatureToPoint_management(temp_join,merged_intersections)
sig_df = CP.arcgis_table_to_df(merged_intersections,["MAXSynchroDataOS
MMatch_Prot","MAXSynchroDataOSMMatch_Perm","MAXSynchroDataOSMMatch_Pro
tPerm","MAXSignalized_Intersection"]).fillna(0)
sig_df.columns = [i.replace("SynchroDataOSMMatch_","").replace("MAX","
") for i in sig_df.columns]
sig_df.sum(axis=0)
```

Out[44]:
```
Prot                        32.0
Perm                         8.0
ProtPerm                    57.0
Signalized_Intersection    166.0
dtype: float64
```