Drill Down, total crime types in the neighborhood West End between january and june

```
SELECT c.crime_type, l.neighborhood, d.month, count(*) AS crimes
FROM crime2 AS c
INNER JOIN fact2 AS f
ON c.crime_key = f.crime_key
INNER JOIN date2 AS d
ON d.date_key = f.date_key
INNER JOIN location2 AS l
ON l.location_key = f.location_key
WHERE neighborhood = 'West End'
AND month BETWEEN 1 AND 6
GROUP BY crime_type
ORDER BY crimes DESC
```

Drill Down, total crime type in Vancouver in the neighborhood West End, in april, at the weekend

```
SELECT c.crime_type, l.city, l.neighborhood, d.month, d.weekend, count(*) AS crimes
FROM crime2 AS c
INNER JOIN fact2 AS f
ON c.crime_key = f.crime_key
INNER JOIN date2 AS d
ON d.date_key = f.date_key
INNER JOIN location2 AS l
ON l.location_key = f.location_key
WHERE city = 'Vancouver'
AND month = 4
AND neighborhood = 'West End'
AND weekend = TRUE
GROUP BY crime_type, city, neighborhood, month, weekend
ORDER BY crimes DESC
```

Drill Down, total fatalities, by hour, in Denver, in december.
It is a drill down operation because we are stepping down the
concept hierarchy for the dimension date and attribute hour

```
SELECT l.city, d.month, d.hour, SUM(f.is_fatal::int) AS fatalities
FROM location2 AS l
INNER JOIN fact2 AS f
ON l.location_key = f.location_key
INNER JOIN date2 AS d
ON d.date_key= f.date_key
WHERE city = 'Denver'
AND month = 12
GROUP BY city, month, hour
ORDER BY fatalities DESC
```

Roll Up, total fatal crimes grouped by crime category and city

```
SELECT c.crime_category, l.city, f.is_fatal, SUM(f.is_fatal::int) AS fatalities
FROM crime2 AS c
INNER JOIN fact2 AS f
ON c.crime_key = f.crime_key
INNER JOIN location2 AS l
ON l.location_key = f.location_key
```

```
GROUP BY crime_category, city, is_fatal
ORDER BY fatalities DESC
```

Roll Up, total crimes grouped by crime category and year

```
SELECT c.crime_category, d.year, count(*) AS crimes
FROM crime2 AS c
INNER JOIN fact2 AS f
ON c.crime_key = f.crime_key
INNER JOIN date2 AS d
ON d.date_key = f.location_key
GROUP BY crime_category, year
ORDER BY crimes DESC
```

Roll Up, total fatalities by year by city.
It is a roll up operation because we are stepping up the
concept hierarchy for the dimension date and attribute year

```
SELECT l.city, d.year, SUM(f.is_fatal::int)
FROM location2 AS l
INNER JOIN fact2 AS f
ON l.location_key = f.location_key
INNER JOIN date2 AS d
ON d.date_key = f.date_key
GROUP BY city, year
ORDER BY sum DESC
```

Slice, total crimes in Vancouver in 2019 grouped by crime category

```
SELECT c.crime_category, l.city, d.year, count(*)
FROM location2 AS l
INNER JOIN fact2 AS f
ON l.location_key = f.location_key
INNER JOIN crime2 AS c
ON c.crime_key = f.crime_key
INNER JOIN date2 as D
ON d.date_key = f.date_key
WHERE city = 'Vancouver'
AND year = 2019
GROUP BY crime_category, city, year
ORDER BY count DESC
```

Slice, total crimes in Vancouver in 2019 grouped by crime type

```
SELECT c.crime_type, l.city, d.year, count(*)
FROM location2 AS l
INNER JOIN fact2 AS f
ON l.location_key = f.location_key
INNER JOIN crime2 AS c
ON c.crime_key = f.crime_key
INNER JOIN date2 as D
ON d.date_key = f.date_key
WHERE city = 'Vancouver'
AND year = 2019
GROUP BY crime_type, city, year
```

```
ORDER BY count DESC
```

Slice, total crimes per city, per crime type, on crime type severity equals 8.
It is a slice operation because we are performing a selection on one dimension
which is the crime dimension and on a specific value for crime severity

```
SELECT l.city, c.crime_type, c.crime_type_severity_index, count(*) AS crimes
FROM crime2 AS c
INNER JOIN fact2 AS f
ON c.crime_key = f.crime_key
INNER JOIN location2 AS l
ON l.location_key = f.location_key
WHERE crime_type_severity_index = 8
GROUP BY city, crime_type, crime_type_severity_index
ORDER BY crimes DESC
```

Dice, total crimes in Vancouver in 2019, 2018 and 2017, grouped by crime type

```
SELECT c.crime_category, l.city, d.year, count(*)
FROM location2 AS l
INNER JOIN fact2 AS f
ON l.location_key = f.location_key
INNER JOIN crime2 AS c
ON c.crime_key = f.crime_key
INNER JOIN date2 as D
ON d.date_key = f.date_key
WHERE city = 'Vancouver'
AND year BETWEEN 2017 AND 2019
GROUP BY crime_category, city, year
ORDER BY year
```

Dice, total crimes in Vancouver and Denver and denver in 2019, grouped by crime type.

```
SELECT c.crime_category, l.city, d.year, count(*)
FROM location2 AS l
INNER JOIN fact2 AS f
ON l.location_key = f.location_key
INNER JOIN crime2 AS c
ON c.crime_key = f.crime_key
INNER JOIN date2 as D
ON d.date_key = f.date_key
WHERE (city = 'Vancouver'
OR city = 'Denver')
AND year = 2019
GROUP BY crime_category, city, year
ORDER BY year DESC, count DESC
```

Dice, total crimes per city, per crime type, per severity index
It is a dice operation because we are performing a selection on
two dimension which are the crime dimension on attribute crime

```
SELECT l.city, c.crime_type, c.crime_type_severity_index, d.year, count(*) AS crimes
FROM crime2 AS c
INNER JOIN fact2 AS f
ON c.crime_key = f.crime_key
```

```
INNER JOIN location2 AS l
ON l.location_key = f.location_key
INNER JOIN date2 AS d
ON d.date_key = f.date_key
WHERE (crime_type_severity_index = 8 OR crime_type_severity_index = 7)
AND (year = 2019 OR year = 2018)
GROUP BY city, crime_type, crime_type_severity_index, year
ORDER BY crimes DESC
```

Combining, total crimes per city per month (from june to december)

```
SELECT l.city, d.month, count(*) AS crimes
FROM location2 AS l
INNER JOIN fact2 AS f
ON l.location_key = f.location_key
INNER JOIN date2 AS d
ON d.date_key = f.date_key
WHERE month BETWEEN 6 AND 12
GROUP BY city, month
ORDER BY crimes DESC
```

Combining, total crimes per city per severity (severities bigger than 3)

```
SELECT l.city, c.crime_type_severity_index, count(*) AS crimes
FROM location2 AS l
INNER JOIN fact2 AS f
ON l.location_key = f.location_key
INNER JOIN crime2 AS c
ON c.crime_key = f.crime_key
WHERE crime_type_severity_index > 3
GROUP BY city, crime_type_severity_index
ORDER BY crimes DESC
```

Combining, total thefts or homicides that happens on both cities during the weekend or weekday, during the year 20
9.
It is a slice query because we are performing on one dimension and on a specific value for year.
It is a dice query because we are performing a selection on the crime dimension on two different
crime categories types.
It is a roll up because we are because we are stepping up concept hierarchy for the dimension
date and attribute year

```
SELECT l.city, c.crime_category, d.weekend, d.year, count(*) AS thefts_or_homicides
FROM location2 AS l
INNER JOIN fact2 AS f
ON l.location_key = f.location_key
INNER JOIN crime2 AS c
ON c.crime_key = f.crime_key
INNER JOIN date2 AS d
ON d.date_key = f.date_key
WHERE (crime_category = 'Theft'
OR crime_category = 'Homicide')
AND year = '2019'
GROUP BY city, crime_category, weekend, year
ORDER BY thefts_or_homicides DESC
```

Iceberg, total crimes in the top 10 neighborhoods in Denver
considering the total population of the neighborhood

```
SELECT l.city, l.neighborhood, l.Total_neighborhood_population, count(*) AS crimes
FROM location2 AS l
INNER JOIN fact2 as f
ON l.location_key = f.location_key
WHERE city = 'Denver'
GROUP BY city, neighborhood, Total_neighborhood_population
ORDER BY crimes DESC
LIMIT 10
```

Iceberg, top 5 most frequent crimes types

```
SELECT c.crime_type, count(*) AS crimes
FROM crime2 as c
GROUP BY crime_type
ORDER BY crimes DESC
LIMIT 5
```

Iceberg, total homicides in the top 10 neighborhoods in Denver.
It is an iceberg operation because it has the constraing "limit to 10"
considering the total population of the neighborhood

```
SELECT l.city, l.neighborhood, l.Total_neighborhood_population, count(*) AS homicides
FROM location2 AS l
INNER JOIN fact2 AS f
ON l.location_key = f.location_key
INNER JOIN crime2 AS c
ON c.crime_key = f.crime_key
WHERE city = 'Denver' AND
crime_category = 'Homicide'
GROUP BY city, neighborhood, Total_neighborhood_population
ORDER BY count DESC
LIMIT 10
```

Windowing on neighborhood, average crime type severity index per city per neighborhood

```
SELECT DISTINCT l.city, l.neighborhood, AVG(c.crime_type_severity_index)
OVER W as avg_severity
FROM location2 AS l
INNER JOIN fact2 AS f
ON l.location_key = f.location_key
INNER JOIN crime2 AS c
ON c.crime_key = f.crime_key
WHERE neighborhood IS NOT NULL
WINDOW w AS (PARTITION BY l.neighborhood)
ORDER BY avg_severity DESC
```

Windowing on city, average crime type severity index per city per neighborhood
It is a windowing operation because it operates on a set of rows, similarly to an aggregate function. However, a window function does not reduce the number of rows returned by the query

```
SELECT DISTINCT l.city, l.neighborhood, AVG(c.crime_type_severity_index)
OVER W as avg_severity
FROM location2 AS l
INNER JOIN fact2 AS f
ON l.location_key = f.location_key
INNER JOIN crime2 AS c
ON c.crime_key = f.crime_key
WHERE neighborhood IS NOT NULL
WINDOW w AS (PARTITION BY l.city)
ORDER BY avg_severity DESC
```

Window on month, average crime type severity index per city per neighborhood

```
SELECT DISTINCT l.city, l.neighborhood, d.month , AVG(c.crime_type_severity_index)
OVER W as avg_severity
FROM location2 AS l
INNER JOIN fact2 AS f
ON l.location_key = f.location_key
INNER JOIN crime2 AS c
ON c.crime_key = f.crime_key
INNER JOIN date2 as d
ON d.date_key = f.date_key
WHERE neighborhood IS NOT NULL
WINDOW w AS (PARTITION BY l.city
ORDER BY d.month
RANGE BETWEEN UNBOUNDED PRECEDING
AND CURRENT ROW)
```

Window on month, average crime type severity index per city per neighborhood
It is a window clause becuase it has a range specified by bounds.

```
SELECT DISTINCT l.city, l.neighborhood, d.month , AVG(c.crime_type_severity_index)
OVER W as avg_severity
FROM location2 AS l
INNER JOIN fact2 AS f
ON l.location_key = f.location_key
INNER JOIN crime2 AS c
ON c.crime_key = f.crime_key
INNER JOIN date2 as d
ON d.date_key = f.date_key
WHERE neighborhood IS NOT NULL
WINDOW w AS (PARTITION BY l.city
ORDER BY d.month
RANGE BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING)
```