Expert Ideation

In this document we shall look into several queries that Geode should be able to answer. Based on human written code to answer the queries, corresponding experts/api requirements shall be gathered.

Query: List three cities near Atlanta, where it will not rain today.

```
def compute_answer(question):
    nearby_cities = proximity_expert('Atlanta', level='city', count=10)
    no_rain_cities = []
    for city in cities:
        prob = rain_prob_expert(point_location_expert(city), date='today')
        if prob < 0.05:
            no_rain_cities.append(city)

    answer_text = elaborate_expert(question=question,
        answer=data_to_text_expert(no_rain_cities))
    return answer_text
```

Experts used: `proximity_expert()`, `rain_prob_expert()`, `point_location_expert()`, `elaborate_expert()`, `data_to_text_expert()`

Query: Find regions with high precipitation and low real estate prices suitable for agriculture in California.

```
def compute_answer(question):
    patch = patch_location_expert('California')

    # in/out patch size is guaranteed to be same,
    # experts can upscale/downscale
    prec_patch = precipitation_expert(patch=patch, point=None)
    price_patch = land_price_expert(patch=patch, point=None)

    # threshold expert returns binary images
    high_prec_patch = threshold_expert(prec_patch, threshold=0.8, mode='greater')
    low_price_patch = threshold_expert(price_patch, threshold=0.2, model='less')

    ideal_patch = high_prec_patch * low_price_patch
    output_map = overlay_expert(base_patch=patch, overlay_patch=ideal_patch)

    return map_output_expert(image=output_map)
```

Experts used: `patch_location_expert()`, `precipitation_expert()`, `land_price_expert()`, `threshold_expert()`, `overlay_expert()`, `map_output_expert()`

Query: Check if there is a correlation between traffic congestion and air quality in Ohio.

```
def compute_answer(question):
    patch = patch_location_expert('Ohio')

    # in/out patch size is guaranteed to be same,
```

```
    # experts can upscale/downscale
    traffic_patch = traffic_expert(patch=patch, point=None)
    aqi_patch = air_quality_expert(patch=patch, point=None)

    imputed_traffic_patch = imputation_expert(patch=traffic_patch)
    correlation = correlation_expert(patch1=traffic_patch, patch2=aqi_patch)

    answer_text = elaborate_expert(question=question,
        answer=data_to_text_expert(correlation))
    return answer_text
```

Experts used: `patch_location_expert()`, `traffic_expert()`, `air_quality_expert()`, `imputation_expert()`, `correlation_expert()`, `elaborate_expert()`, `data_to_text_expert()`

Query: Which country has the larger area, Greenland or Russia?

```
def compute_answer(question):
    greenland_patch = patch_location_expert('Greenland')
    russia_patch = patch_location_expert('Russia')

    # patch has access to both boundary and image information
    greenland_area = get_area(greenland_patch)
    russia_area = get_area(russia_patch)
    if russia_area > greenland_area:
        answer_text = elaborate_expert(question=question,
            answer=f'Russia has larger area ({russia_area})')
    else:
        answer_text = elaborate_expert(question=question,
            answer=f'Greenland has larger area ({greenland_area})')
    return answer_text
```

Experts used: `patch_location_expert()`, `get_area()`, `elaborate_expert()`

Query: Which country has the larger area, Greenland or Russia?

```
def compute_answer(question):
    greenland_patch = patch_location_expert('Greenland')
    russia_patch = patch_location_expert('Russia')

    # patch has access to both boundary and image information
    greenland_area = get_area(greenland_patch)
    russia_area = get_area(russia_patch)
    if russia_area > greenland_area:
        answer_text = elaborate_expert(question=question,
            answer=f'Russia has larger area ({russia_area})')
    else:
        answer_text = elaborate_expert(question=question,
            answer=f'Greenland has larger area ({greenland_area})')
    return answer_text
```

Experts used: `patch_location_expert()`, `get_area()`, `elaborate_expert()`

Query: Which city out of San Fransisco, New York and Washington DC is the most affordable in terms of income and housing?

```
def compute_answer(question):
    cities = ['San Fransisco', 'New York', 'Washington DC']
    patches = [patch_location_expert(city) for city in cities]

    avg_incomes = [income_expert(city) for city in cities]

    price_patches = [land_price_expert(patch=patch) for patch in patches]
    avg_prices = [np.mean(price_patch) for price_patch in price_patches]

    affordability = [inc/price for inc, price in zip(avg_incomes, avg_prices)]
    best_idx = np.argmax(affordability)
    most_affordable_city = cities[best_idx]

    answer_text = elaborate_expert(question=question,
            answer=f'Most affordable city {most_affordable_city}',
            context=[f'ratio of income to land price = {affordability}',
                f'Average incomes = {avg_incomes}',
                f'Average land prices = {avg_prices}',
                f'Ratios of income and land prices = {affordability}'])

    return answer_text
```

Experts used: `patch_location_expert()`, `income_expert()`, `land_price_expert()`, `elaborate_expert()`