FINAL REPORT

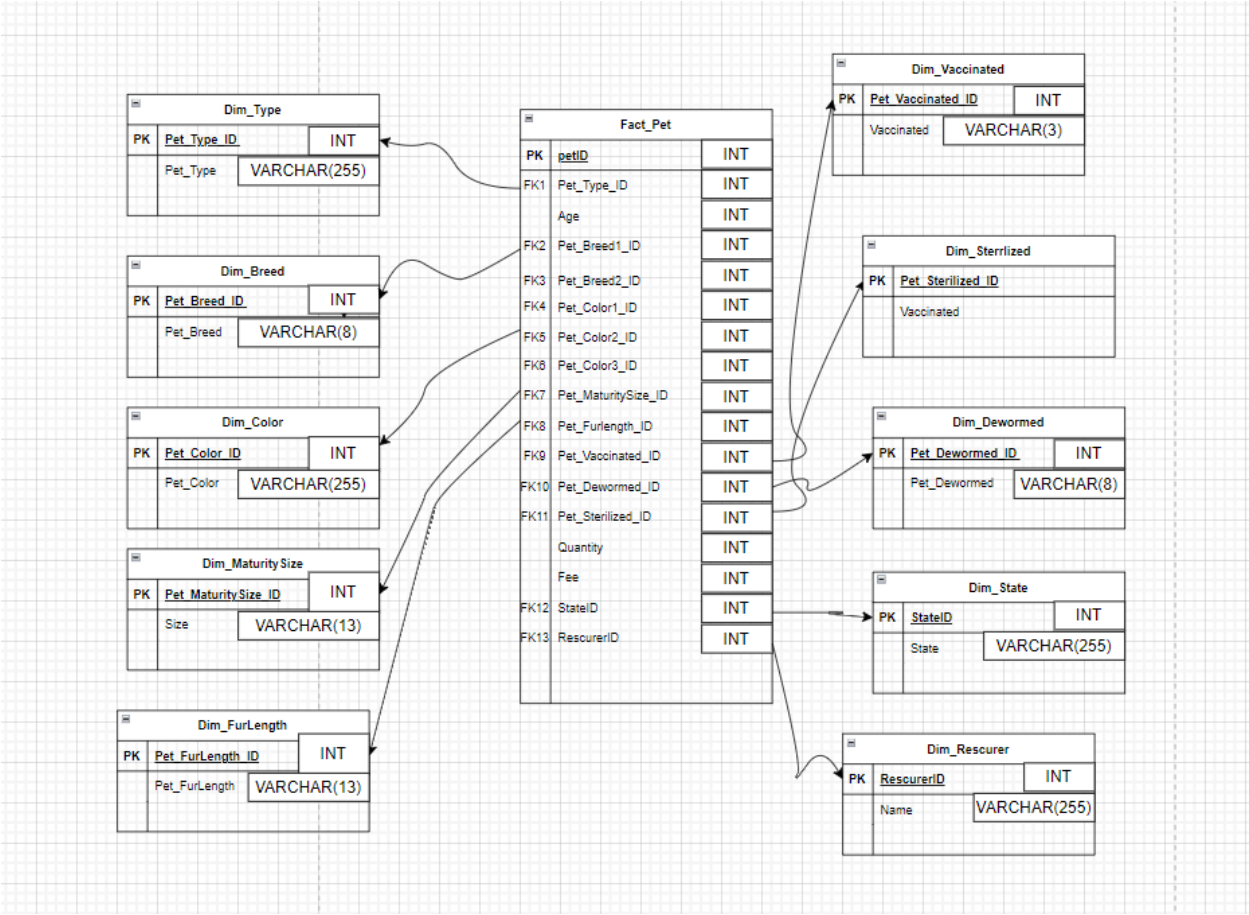
1) Business problems

The PetFinder data file contains information about pets, including the following information: Name, age, breed, coat color, etc. Our primary task includes building a data warehouse for the PetFinder data file following below requirements:

1. Entity-Relationship Diagram to visualize the relational database
2. ETL Pipelines constructing in SSIS
3. SQL File constructing the database
4. Business queries to test data warehouse efficiency

2) Entity-Relationship Diagram

I will normalize the PetFinder dataset using star schema design, represented by the picture (picture1) below. One key advantage of the star schema is its simplicity and ease of query performance optimization. By organizing data into a central fact table that contains quantitative measures and surrounding it with dimension tables that provide context, the star schema facilitates efficient data retrieval and analysis. Additionally, the star schema's straightforward design makes it easier for business users to understand and navigate, fostering a more user-friendly and intuitive environment for data exploration and reporting.All of the dimensional tables' primary keys, except Dim_Rescurer, are surrogate keys, using unique identifiers when incrementally load each row into a certain table.



***Picture1**: Representing the implemented data warehouse structure.*

***Pet_Fact:** Primary (Fact) table contains pet information*

***Dim_Type:** Dimension table contains pet's type information*

***Dim_Breed:** Dimension (Fact) table contains pet's breed information*

***Dim_Color:** Dimension (Fact) table contains pet's color information*

***Dim_MaturitySize:** Dimension (Fact) table contains pet's maturity size information*

***Dim_FurLength:** Dimension (Fact) table contains pet's fur length information*

***Dim_Vaccinated:** Dimension (Fact) table contains pet's vaccination condition information*

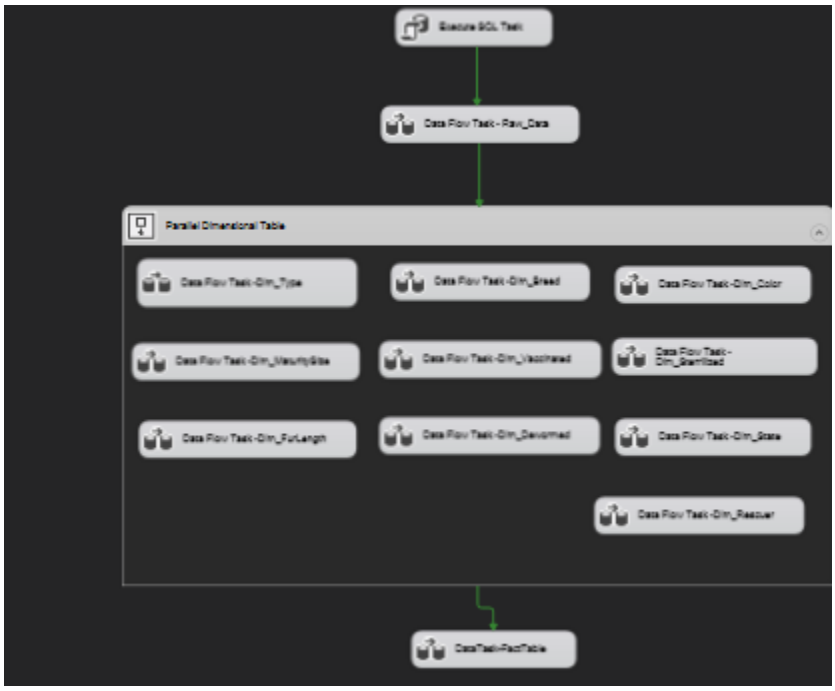
***Dim_Dewormed:** Dimension (Fact) table contains pet's dewormed condition information*

***Dim_Sterilized:** Dimension (Fact) table contains pet's sterilized condition information*

***Dim_State:** Dimension (Fact) table contains pet's state information*

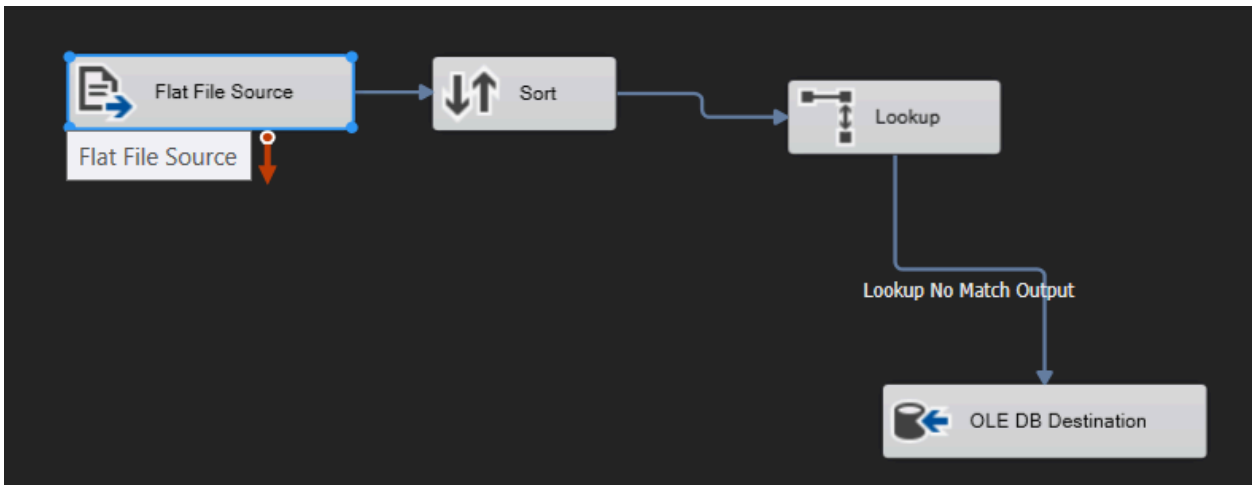
***Dim_Rescurer:** Dimension (Fact) table contains rescurer's information*

3) ETL Design

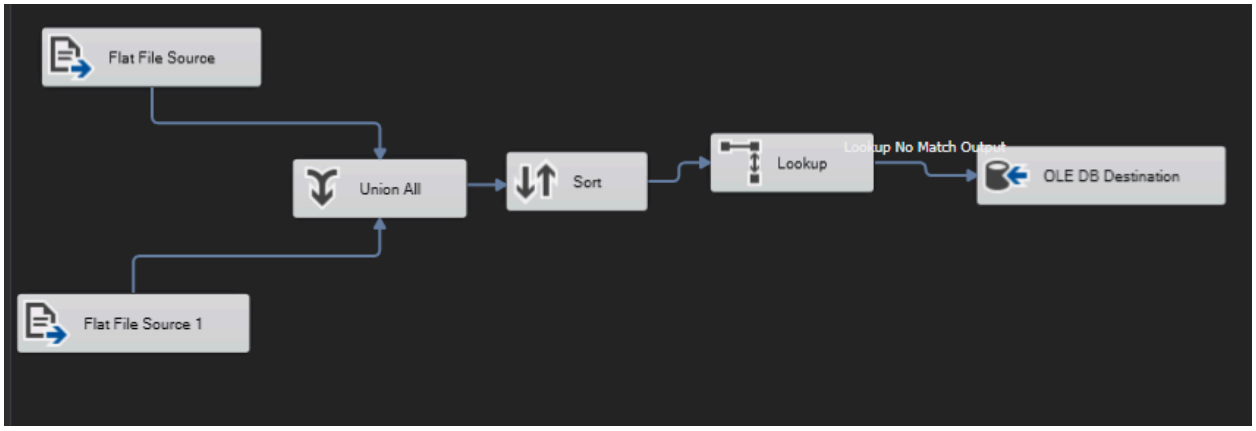


Picture2: At first, we will execute an SQL file to create an empty data warehouse. Then. The dimensional containers, including 12 ETL pipelines for dimensional tables, will run and load data into all of the dimensional tables. Finally, the ETL pipeline for fact table will be implemented.

Dimensional Containers: 12 ETL data flows running parallely in the container and ultimately, the data flow task for the fact table will be executed. Since all foreign keys in the fact table depend on the primary (surrogate) key in all dimensional tables, SSIS will apply lookup capability to perform switching these values into lookup id inside the fact table. Hence, dimensional ETLs will be constructed from these 2 designs:

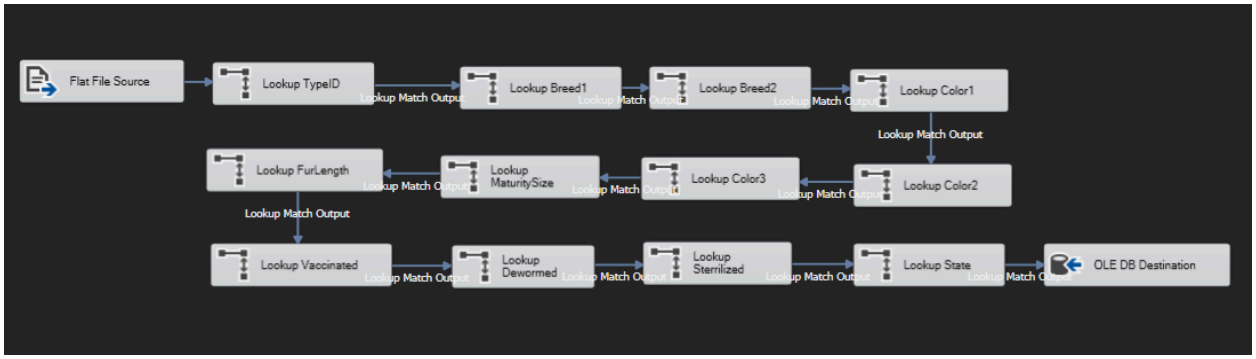


Picture3: ETL pipeline for singular column



Picture4: ETL pipeline for merging columns into a single column (breed1 and breed2 into breed)

Finally, this is the pipeline for fact table:



Picture5: The ETL pipeline includes multiple lookups to generate foreign keys by matching each key-value set.

4) Business Query

a) Comparison between dogs and cats

Query:

```
-- 1) Dogs vs Cats
SELECT
    T.Pet_Type,
    COUNT(F.PetID ) AS Total_Case,
    SUM(F.Quantity) AS Total_Number,
    SUM(F.Fee)/SUM(F.Quantity) AS Average_Price,
    MAX(F.age) AS maximum_age,
    MIN(F.age) AS minimum_age,
    ROUND(COUNT(CASE WHEN V.Pet_Vaccinated = 'Yes' THEN F.PetID ELSE NULL END) * 1.0/COUNT(F.PetID),2) AS pct_vaccinated,
    ROUND(COUNT(CASE WHEN D.Pet_Dewormed = 'Yes' THEN F.PetID ELSE NULL END) * 1.0/COUNT(F.PetID),2) AS pct_dewormed,
    ROUND(COUNT(CASE WHEN S.Pet_Sterilized = 'Yes' THEN F.PetID ELSE NULL END) * 1.0/COUNT(F.PetID),2) AS pct_sterilized
FROM Fact_Pet F
LEFT JOIN Dim_Vaccinated V ON F.Pet_Vaccinated_ID = V.Pet_Vaccinated_ID
LEFT JOIN Dim_Dewormed D ON F.Pet_Dewormed_ID = D.Pet_Dewormed_ID
LEFT JOIN Dim_Sterilized S ON F.Pet_Sterilized_ID = S.Pet_Sterlized_ID
LEFT JOIN Dim_Type T ON T.Pet_Type_ID = F.Pet_Type_ID
GROUP BY T.Pet_Type;
```

Result:

Results | Messages

	Pet_Type	Total_Case	Total_Number	Average_Price	maximum_age	minimum_age	pct_vaccinated	pct_dewormed	pct_sterilized
1	Cat	6861	11213	10	212	0	0.290000000000	0.510000000000	0.180000000000
2	Dog	8132	12414	15	255	0	0.480000000000	0.600000000000	0.230000000000

b) Differences between breed

Primary Breed:

Query:

```
SELECT
    p1.Pet_Breed,
    COUNT(DISTINCT f.PetID) AS num_case,
    SUM(F.Quantity) AS Total_Number,
    SUM(F.Fee)/MAX(F.Quantity) AS Avg_Price,
    MAX(F.age) AS maximum_age,
    MIN(F.age) AS minimum_age,
    ROUND(COUNT(CASE WHEN V.Pet_Vaccinated = 'Yes' THEN F.PetID ELSE NULL END) * 1.0/COUNT(F.PetID),2) AS pct_vaccinated,
    ROUND(COUNT(CASE WHEN D.Pet_Dewormed = 'Yes' THEN F.PetID ELSE NULL END) * 1.0/COUNT(F.PetID),2) AS pct_dewormed,
    ROUND(COUNT(CASE WHEN S.Pet_Sterilized = 'Yes' THEN F.PetID ELSE NULL END) * 1.0/COUNT(F.PetID),2) AS pct_sterilized
FROM Fact_Pet f
LEFT JOIN Dim_Breed p1 ON f.Pet_Breed1_ID = p1.Pet_Breed_ID
LEFT JOIN Dim_Vaccinated V ON F.Pet_Vaccinated_ID = V.Pet_Vaccinated_ID
LEFT JOIN Dim_Dewormed D ON F.Pet_Dewormed_ID = D.Pet_Dewormed_ID
LEFT JOIN Dim_Sterilized S ON F.Pet_Sterilized_ID = S.Pet_Sterlized_ID
GROUP BY p1.Pet_Breed;
```

Result:

	Pet_Breed	num_case	Total_Number	Avg_Price	maximum_age	minimum_age	pct_vaccinated	pct_dewormed	pct_sterilized
1	Belgian Shepherd Laekenois	3	4	0	7	3	0.330000000000	0.670000000000	0.000000000000
2	American Shorthair	94	159	15	108	0	0.190000000000	0.300000000000	0.200000000000
3	Doberman Pinscher	62	74	43	120	1	0.760000000000	0.740000000000	0.400000000000
4	Lowchen	1	1	0	2	2	0.000000000000	1.000000000000	0.000000000000
5	German Pinscher	6	7	0	29	3	0.170000000000	0.170000000000	0.000000000000
6	Tabby	342	560	10	80	0	0.310000000000	0.560000000000	0.230000000000
7	Retriever	4	6	0	84	2	0.750000000000	0.750000000000	0.000000000000
8	Chihuahua	37	46	95	120	2	0.730000000000	0.700000000000	0.300000000000
9	Tonkinese	5	6	0	144	4	0.800000000000	0.800000000000	0.800000000000
10	Dalmatian	39	52	46	144	1	0.670000000000	0.770000000000	0.380000000000
11	Kuvasz	1	2	0	8	8	1.000000000000	1.000000000000	0.000000000000
12	Cymric	2	2	0	14	11	0.000000000000	0.500000000000	0.000000000000
13	Foxhound	1	1	0	24	24	1.000000000000	1.000000000000	1.000000000000
14	Shih Tzu	190	207	87	122	1	0.690000000000	0.730000000000	0.330000000000
15	Silver	4	4	50	9	5	1.000000000000	1.000000000000	0.750000000000
16	Miniature Pinscher	67	75	34	120	1	0.520000000000	0.730000000000	0.270000000000
17	Manx	8	12	12	60	1	0.380000000000	0.630000000000	0.130000000000
18	Birman	2	2	5	28	0	1.000000000000	1.000000000000	0.500000000000
19	Irish Wolfhound	1	1	0	10	10	1.000000000000	1.000000000000	1.000000000000
20	Fox Terrier	2	2	0	24	2	0.000000000000	0.000000000000	0.000000000000
21	Pug	21	22	71	120	7	0.810000000000	0.860000000000	0.570000000000
22	Egyptian Mau	3	4	33	9	4	0.670000000000	0.330000000000	0.330000000000
23	Irish Terrier	1	1	200	3	3	1.000000000000	1.000000000000	0.000000000000
24	Maltese	18	19	130	84	2	0.940000000000	0.940000000000	0.440000000000
25	Havana	2	2	0	12	10	0.500000000000	0.000000000000	0.500000000000
26	Singapura	13	27	3	12	1	0.310000000000	0.460000000000	0.230000000000
27	Pit Bull Terrier	23	49	55	30	0	0.570000000000	0.650000000000	0.350000000000
28	Afghan Hound	3	3	1	2	1	0.330000000000	0.000000000000	0.330000000000
29	Burmese	23	28	15	78	0	0.480000000000	0.650000000000	0.430000000000
30	Akita	2	2	0	24	0	0.500000000000	0.500000000000	0.500000000000
31	Jack Russell Terrier	64	72	46	132	0	0.800000000000	0.800000000000	0.420000000000
32	Whippet	2	2	0	18	5	1.000000000000	1.000000000000	0.500000000000
33	Standard Poodle	2	2	0	51	28	1.000000000000	1.000000000000	0.000000000000
34	Belgian Shepherd Malinois	26	33	42	84	2	0.730000000000	0.620000000000	0.190000000000
35	Shiba Inu	3	3	83	36	3	0.670000000000	0.670000000000	0.330000000000
36	Australian Terrier	6	8	16	84	4	0.670000000000	1.000000000000	0.170000000000

Secondary Breed:

Query:

```
SELECT
    p2.Pet_Breed,
    COUNT(DISTINCT f.PetID) AS num_case,
    SUM(F.Quantity) AS Total_Number,
    AVG(F.Fee)/AVG(F.Quantity) AS Avg_Price,
    MAX(F.age)/AVG(F.Quantity) AS maximum_age,
    MIN(F.age)/AVG(F.Quantity) AS minimum_age,
    ROUND(COUNT(CASE WHEN V.Pet_Vaccinated = 'Yes' THEN F.PetID ELSE NULL END) * 1.0/COUNT(F.PetID),2) AS pct_vaccinated,
    ROUND(COUNT(CASE WHEN D.Pet_Dewormed = 'Yes' THEN F.PetID ELSE NULL END) * 1.0/COUNT(F.PetID),2) AS pct_dewormed,
    ROUND(COUNT(CASE WHEN S.Pet_Sterilized = 'Yes' THEN F.PetID ELSE NULL END) * 1.0/COUNT(F.PetID),2) AS pct_sterilized
FROM Fact_Pet f
LEFT JOIN Dim_Breed p2 ON f.Pet_Breed2_ID = p2.Pet_Breed_ID
LEFT JOIN Dim_Vaccinated V ON F.Pet_Vaccinated_ID = V.Pet_Vaccinated_ID
LEFT JOIN Dim_Dewormed D ON F.Pet_Dewormed_ID = D.Pet_Dewormed_ID
LEFT JOIN Dim_Sterilized S ON F.Pet_Sterilized_ID = S.Pet_Sterlized_ID
GROUP BY p2.Pet_Breed;
```

Result:

	Pet_Breed	num_case	Total_Number	Avg_Price	maximum_age	minimum_age	pct_vaccinated	pct_dewormed	pct_sterilized
1	Belgian Shepherd Laekenois	1	1	0	7	7	1.000000000000	1.000000000000	0.000000000000
2	American Shorthair	30	49	10	51	0	0.270000000000	0.500000000000	0.230000000000
3	Doberman Pinscher	22	28	51	20	1	0.730000000000	0.820000000000	0.450000000000
4	Border Terrier	1	1	0	36	36	0.000000000000	0.000000000000	0.000000000000
5	Selkirk Rex	1	1	0	7	7	0.000000000000	0.000000000000	0.000000000000
6	Lowchen	1	1	0	2	2	0.000000000000	1.000000000000	0.000000000000
7	German Pinscher	2	2	0	9	3	0.500000000000	0.500000000000	0.500000000000
8	Tabby	138	210	15	60	0	0.270000000000	0.560000000000	0.190000000000
9	Retriever	9	13	33	84	2	0.440000000000	0.670000000000	0.000000000000
10	Chihuahua	7	7	0	36	3	0.710000000000	0.710000000000	0.430000000000
11	Tonkinese	1	1	0	5	5	0.000000000000	0.000000000000	0.000000000000
12	Dalmatian	14	22	7	18	2	0.640000000000	0.500000000000	0.290000000000
13	Cymric	1	1	100	2	2	1.000000000000	1.000000000000	0.000000000000