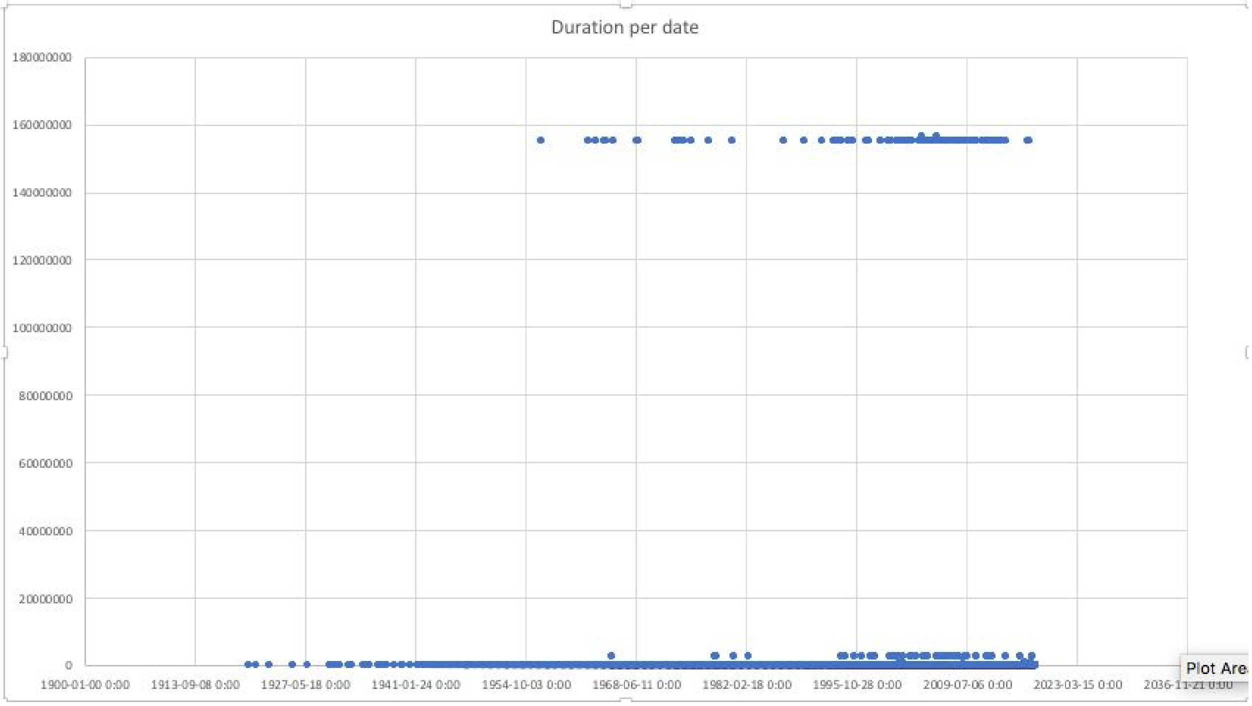
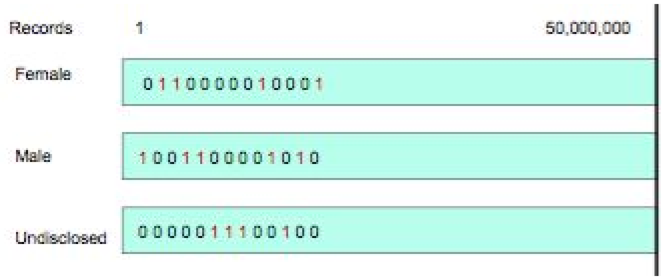
Felix Singerman
7970742
February 12, 2018

Assignment 1

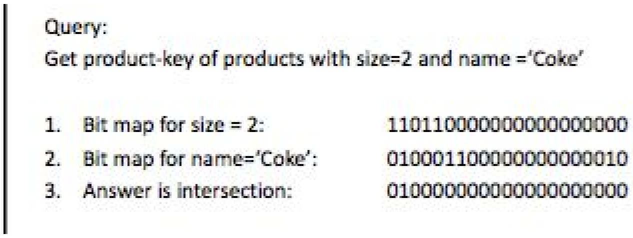
1. The grain of the model is: UFO's and its shape reported by location and date.
2. The mission dimension is Event-Date. The CSV has a posted date and the date that the UFO sighting took place. The current diagram only had the posting date. It is vital to have the information of when the actual UFO sighting took place.
3. Each aggregate occupies its own fact table. A fact is a business measure. A fact table is where numeric performance measures for a business process are stored. Facts are normally numeric and additive. They have to answer the question "What are we measuring?" I would add a measure of which city has the most UFO sightings.
4. A concept hierarchy is a summarization of data by climbing up the hierarchy. It groups values together for a certain attribute. It follows the parent-child relationship. In our UFO data mart we can have a concept hierarchy on Location. The Location-Dimension is broken down into region, countries, state, and cities.
5. I would create an aggregate in order to speed up frequent queries on counting UFO sightings as a function of City, shape, and month.
6. An attribute that has missing values is the Duration attribute. In some places there is a range of duration (ie: 10-20 seconds), in some places the duration is completely missing, and in some places the unit of measurement is different. I would handle this by combining the data that has a duration and not let the data that doesn't have a duration affect the results by excluding it from any duration calculation as to not let it skew the data. I will also normalize the data so that everything was in seconds and that we would take the median of each range.

7. 

8. Using bitmap indexes is good if the cardinality is small. We would have a bitmap for the states and and a bitmap delta based shaped. We would then use the intersection of the 2 bitmaps in order to find out results. Similar to the lecture notes in class however instead of male and female we would have the state names each corresponding to their own bitmap. As well as having the delta shapes corresponding to their own bitmap. FInally we will take the intersection of all of them and we would get our result.



Records	1	50,000,000
Female	0110000010001	
Male	1001100001010	
Undisclosed	0000011100100	



Query:
Get product-key of products with size=2 and name ='Coke'

1. Bit map for size = 2: 11011000000000000000
2. Bit map for name='Coke': 01000110000000000010
3. Answer is intersection: 01000000000000000000

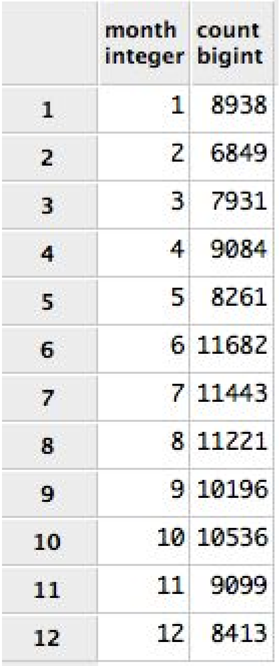
9. I would have all the UFO sightings from 2017 as our first input and check to see if they were diamond. Our bitmap index would only be comparing sightings of 2017 and not the entire database. We would then run an intersection on if it is a diamond and the year 2017 as we did in Q8.

10. Data mart created in postgresql.

```
-- create tables
CREATE TABLE public."Event-Date"
(
  "event-date-key" integer NOT NULL DEFAULT
nextval('"Event-Date_event-date-key_seq"'::regclass),
  date date,
  week integer,
  month integer,
  year integer,
  weekend text,
  season text,
  "special-event" text,
  CONSTRAINT "Event-Date_pkey" PRIMARY KEY ("event-date-key")
)
WITH (
  OIDS=FALSE
);
CREATE TABLE public."Location"
(
  city text,
  state text,
  country text,
  region text,
  "location-key" integer NOT NULL DEFAULT nextval('"Location_location-key_seq"'::regclass),
  CONSTRAINT "Location_pkey" PRIMARY KEY ("location-key")
)
CREATE TABLE public."Reported-Date"
(
  date date,
  week text,
  month text,
  year integer,
  weekend text,
  season text,
  "specialEvent" text,
  "reported-date-key" integer NOT NULL DEFAULT
nextval('"Reported-Date_reported-date-key_seq"'::regclass),
  CONSTRAINT "Reported-Date_pkey" PRIMARY KEY ("reported-date-key")
)
CREATE TABLE public."Shape"
(
  "shape-name" text,
  summary text,
  "shape-key" integer NOT NULL DEFAULT nextval('"Shape_shape-key_seq"'::regclass),
  CONSTRAINT "Shape_pkey" PRIMARY KEY ("shape-key")
)
CREATE TABLE public."UFO-fact"
(
  "reported-date-key" integer NOT NULL,
  "shape-key" integer NOT NULL,
  "location-key" integer NOT NULL,
  "event-day-key" integer NOT NULL,
  duration integer,
  CONSTRAINT "UFO-fact_pkey" PRIMARY KEY ("reported-date-key", "shape-key",
"location-key", "event-day-key"),
  CONSTRAINT l FOREIGN KEY ("location-key")
      REFERENCES public."Location" ("location-key") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT s FOREIGN KEY ("shape-key")
      REFERENCES public."Shape" ("shape-key") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT t FOREIGN KEY ("event-day-key")
      REFERENCES public."Event-Date" ("event-date-key") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT v FOREIGN KEY ("reported-date-key")
      REFERENCES public."Reported-Date" ("reported-date-key") MATCH SIMPLE
      ON UPDATE CASCADE ON DELETE CASCADE
)

-- import the data from csv
COPY "ufo"("EventDate","Location", "Province", "Shape", "Duration", "Description",
"DatePosted")
FROM
'/Users/FLSingerman/Documents/Ottawa_U/Year_4_2017_18/Winter/Data_Science/Assignment
s/FINN.csv' DELIMITER ',' CSV HEADER;

-- input data into corresponding tables
Insert into "Location"(city,state,"location-key")
Select "Location", "Province", "id"
FROM UFO;

Insert into "Shape"("shape-name",summary,"shape-key")
Select "Shape", "Description", "id"
FROM UFO;
```

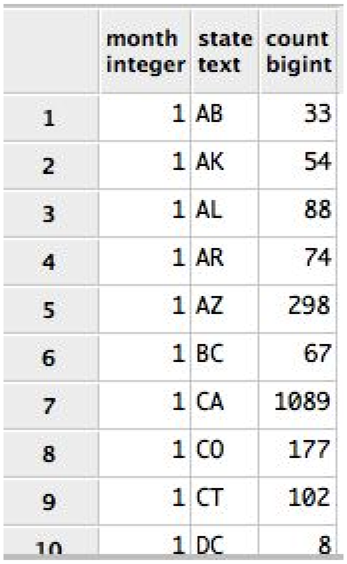
```
Insert into "Event-Date"(date,week,month,year,weekend,"event-date-key")
Select "EventDate", EXTRACT(week FROM "EventDate") as week, EXTRACT(MONTH FROM
"EventDate") as Month, EXTRACT(YEAR FROM "EventDate") as Year,
        CASE
        WHEN to_char("EventDate", 'D') = '7' THEN 'y'
        WHEN to_char("EventDate", 'D') = '1' then 'y'
        ELSE 'n'
        END,
        "id"
FROM UFO;

Insert into "Reported-Date"(date,week,month,year,weekend,"reported-date-key")
Select "DatePosted", EXTRACT(week FROM "DatePosted") as week, EXTRACT(MONTH
FROM "DatePosted") as Month, EXTRACT(YEAR FROM "DatePosted") as Year,
        CASE
        WHEN to_char("DatePosted", 'D') = '7' THEN 'y'
        WHEN to_char("DatePosted", 'D') = '1' then 'y'
        ELSE 'n'
        END,
        "id"
FROM UFO;

Insert into "UFO-fact"("event-day-key","location-key","shape-key","reported-date-key","duration")
Select e."event-date-key" , l."location-key", s."shape-key", r."reported-date-key", u."Duration"
FROM "Location" l, "Shape" s, "Reported-Date" r, "Event-Date" e, "ufo" u
WHERE e."event-date-key" = u."id" and u."id" = l."location-key" and u."id" =
r."reported-date-key" and u."id" = s."shape-key";
```

```
11)     Select month, count("month")
        FROM "Event-Date"
        GROUP BY Month Order BY Month;
```



	month integer	count bigint
1	1	8938
2	2	6849
3	3	7931
4	4	9084
5	5	8261
6	6	11682
7	7	11443
8	8	11221
9	9	10196
10	10	10536
11	11	9099
12	12	8413

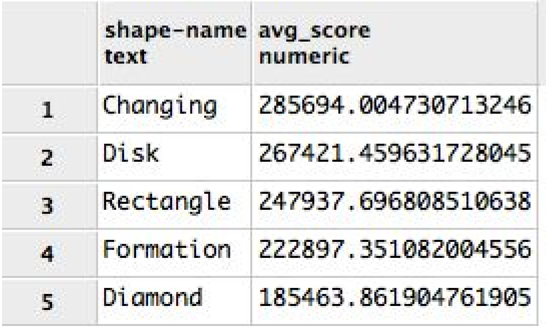
12)
```
SELECT e.month,  l.state, count(e.month)
FROM "Event-Date" e, "Location" l, "UFO-fact" u
WHERE (e."event-date-key" = u."event-day-key") AND  (l."location-key" =
u."location-key" )
GROUP BY e.month, l.state Order BY e.month, l.state;
```



	month integer	state text	count bigint
1	1	AB	33
2	1	AK	54
3	1	AL	88
4	1	AR	74
5	1	AZ	298
6	1	BC	67
7	1	CA	1089
8	1	CO	177
9	1	CT	102
10	1	DC	8

13)
```
Select  s."shape-name", AVG(u."duration") as avg_score
FROM "Shape" s, "UFO-fact" u
WHERE s."shape-key" = u."shape-key"
```

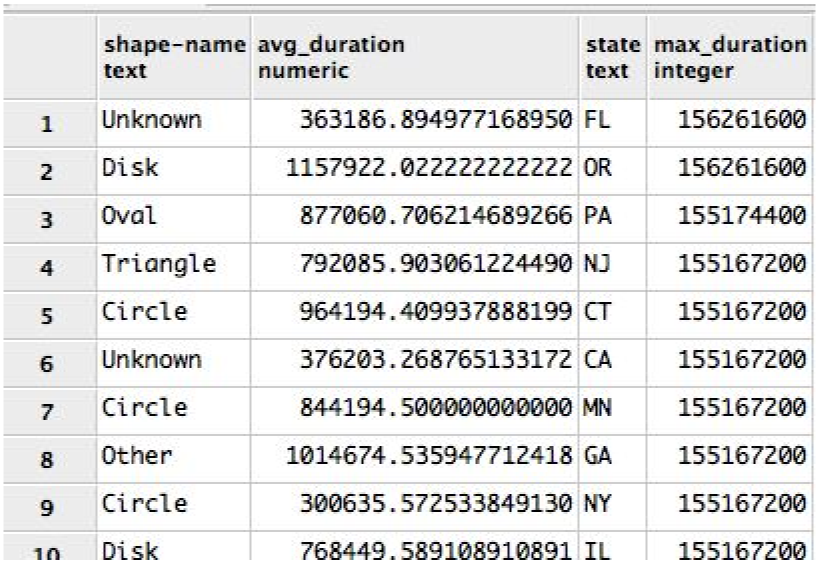
```
GROUP BY s."shape-name"  ORDER BY avg_score desc
LIMIT 5 ;
```



	shape-name text	avg_score numeric
1	Changing	285694.004730713246
2	Disk	267421.459631728045
3	Rectangle	247937.696808510638
4	Formation	222897.351082004556
5	Diamond	185463.861904761905

14)

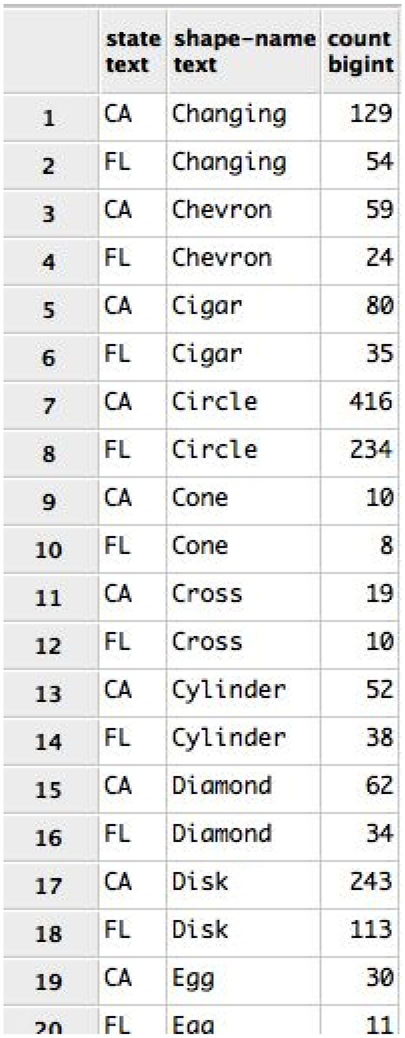
```
SELECT s."shape-name", AVG(u."duration") as avg_duration, l."state",
MAX(u."duration") as max_duration
FROM "Shape" s, "UFO-fact" u, "Location" l
WHERE s."shape-key" = u."shape-key" AND l."location-key" = u."location-key"
GROUP BY l."state", s."shape-name"  ORDER BY max_duration desc;
```



	shape-name text	avg_duration numeric	state text	max_duration integer
1	Unknown	363186.894977168950	FL	156261600
2	Disk	1157922.022222222222	OR	156261600
3	Oval	877060.706214689266	PA	155174400
4	Triangle	792085.903061224490	NJ	155167200
5	Circle	964194.409937888199	CT	155167200
6	Unknown	376203.268765133172	CA	155167200
7	Circle	844194.500000000000	MN	155167200
8	Other	1014674.535947712418	GA	155167200
9	Circle	300635.572533849130	NY	155167200
10	Disk	768449.589108910891	IL	155167200

15)

```
SELECT l."state", s."shape-name", COUNT(s."shape-name")
FROM "Location" l, "Event-Date" e, "Shape" s, "UFO-fact" u
WHERE (l."state" = 'CA' OR l."state" = 'FL') AND s."shape-key" = u."shape-key" AND
l."location-key" = u."location-key" AND e."event-date-key" = u."event-day-key" AND
e."weekend" = 'y'
GROUP BY s."shape-name", l."state" ORDER BY s."shape-name", l."state";
```



	state text	shape-name text	count bigint
1	CA	Changing	129
2	FL	Changing	54
3	CA	Chevron	59
4	FL	Chevron	24
5	CA	Cigar	80
6	FL	Cigar	35
7	CA	Circle	416
8	FL	Circle	234
9	CA	Cone	10
10	FL	Cone	8
11	CA	Cross	19
12	FL	Cross	10
13	CA	Cylinder	52
14	FL	Cylinder	38
15	CA	Diamond	62
16	FL	Diamond	34
17	CA	Disk	243
18	FL	Disk	113
19	CA	Egg	30
20	FL	Egg	11