1. Exploring natality dataset

This notebook illustrates:

1. Exploring a BigQuery dataset using AI Platform Notebooks.

In [1]:
```
!sudo chown -R jupyter:jupyter /home/jupyter/training-data-analyst
```

In [2]:
```
# change these to try this notebook out
BUCKET = 'cloud-training-demos-ml'
PROJECT = 'cloud-training-demos'
REGION = 'us-central1'
```

In [3]:
```
import os
os.environ['BUCKET'] = BUCKET
os.environ['PROJECT'] = PROJECT
os.environ['REGION'] = REGION
```

In [5]:
```
%%bash
if ! gsutil ls | grep -q gs://${BUCKET}/; then
  gsutil mb -l ${REGION} gs://${BUCKET}
fi
```

```
Creating gs://cloud-training-demos-ml/...
ServiceException: 409 A Cloud Storage bucket named 'cloud-training-demos-ml' alr
eady exists. Try another name. Bucket names must be globally unique across all G
oogle Cloud projects, including those outside of your organization.

---------------------------------------------------------------------------
CalledProcessError                        Traceback (most recent call last)
<ipython-input-5-6b1d45d375e6> in <module>
----> 1 get_ipython().run_cell_magic('bash', '', 'if ! gsutil ls | grep -q gs://
${BUCKET}/; then\n  gsutil mb -l ${REGION} gs://${BUCKET}\nfi\n')

/opt/conda/lib/python3.7/site-packages/IPython/core/interactiveshell.py in run_c
ell_magic(self, magic_name, line, cell)
   2401             with self.builtin_trap:
   2402                 args = (magic_arg_s, cell)
-> 2403                 result = fn(*args, **kwargs)
   2404             return result
   2405

/opt/conda/lib/python3.7/site-packages/IPython/core/magics/script.py in named_sc
ript_magic(line, cell)
    140             else:
    141                 line = script
--> 142             return self.shebang(line, cell)
    143
    144         # write a basic docstring:

/opt/conda/lib/python3.7/site-packages/decorator.py in fun(*args, **kw)
    230             if not kwsyntax:
    231                 args, kw = fix(args, kw, sig)
```

```
--> 232             return caller(func, *(extras + args), **kw)
    233         fun.__name__ = func.__name__
    234         fun.__doc__ = func.__doc__

/opt/conda/lib/python3.7/site-packages/IPython/core/magic.py in <lambda>(f, *a,
 **k)
    185         # but it's overkill for just that one bit of state.
    186         def magic_deco(arg):
--> 187             call = lambda f, *a, **k: f(*a, **k)
    188 
    189             if callable(arg):

/opt/conda/lib/python3.7/site-packages/IPython/core/magics/script.py in shebang
(self, line, cell)
    243                 sys.stderr.flush()
    244             if args.raise_error and p.returncode!=0:
--> 245                 raise CalledProcessError(p.returncode, cell, output=out, std
err=err)
    246 
    247         def _run_script(self, p, cell, to_close):

CalledProcessError: Command 'b'if ! gsutil ls | grep -q gs://${BUCKET}/; then\n
gsutil mb -l ${REGION} gs://${BUCKET}\nfi\n'' returned non-zero exit status 1.
```

Explore data

The data is natality data (record of births in the US). My goal is to predict the baby's weight given a number of factors about the pregnancy and the baby's mother. Later, we will want to split the data into training and eval datasets. The hash of the year-month will be used for that -- this way, twins born on the same day won't end up in different cuts of the data.

In [6]:

```
# Create SQL query using natality data after the year 2000
query = """
SELECT
  weight_pounds,
  is_male,
  mother_age,
  plurality,
  gestation_weeks,
  FARM_FINGERPRINT(CONCAT(CAST(YEAR AS STRING), CAST(month AS STRING))) AS hashm
FROM
  publicdata.samples.natality
WHERE year > 2000
"""
```

In [7]:

```
# Call BigQuery and examine in dataframe
from google.cloud import bigquery
df = bigquery.Client().query(query + " LIMIT 100").to_dataframe()
df.head()
```

Out[7]:

	weight_pounds	is_male	mother_age	plurality	gestation_weeks	hashmonth
0	7.063611	True	32	1	37.0	7108882242435606404
1	4.687028	True	30	3	33.0	-7170969733900686954

	weight_pounds	is_male	mother_age	plurality	gestation_weeks	hashmonth
2	7.561856	True	20	1	39.0	6392072535155213407
3	7.561856	True	31	1	37.0	-2126480030009879160
4	7.312733	True	32	1	40.0	3408502330831153141

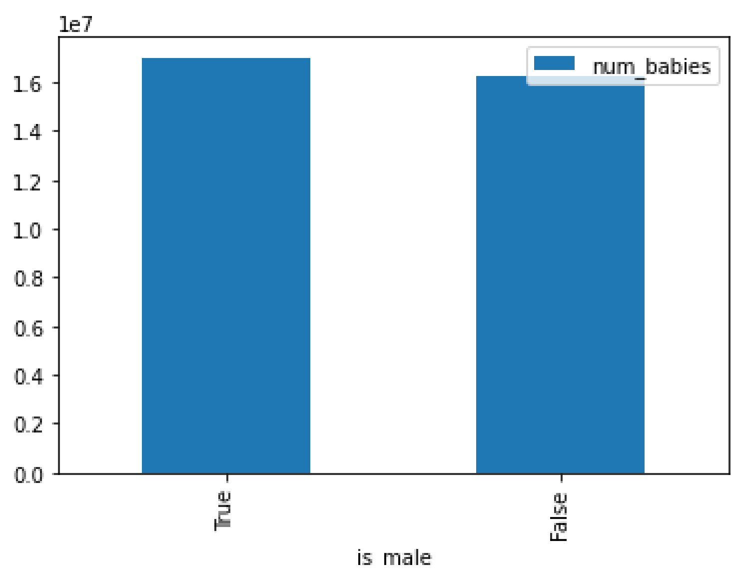
Let's write a query to find the unique values for each of the columns and the count of those values. This is important to ensure that we have enough examples of each data value, and to verify our hunch that the parameter has predictive value.

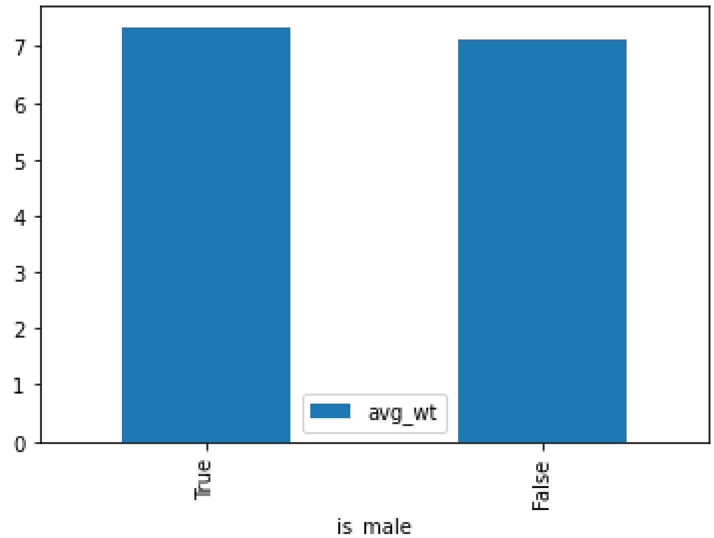
In [8]:

```
# Create function that finds the number of records and the average weight for ea
def get_distinct_values(column_name):
  sql = """
SELECT
  {0},
  COUNT(1) AS num_babies,
  AVG(weight_pounds) AS avg_wt
FROM
  publicdata.samples.natality
WHERE
  year > 2000
GROUP BY
  {0}
  """.format(column_name)
  return bigquery.Client().query(sql).to_dataframe()
```

In [9]:

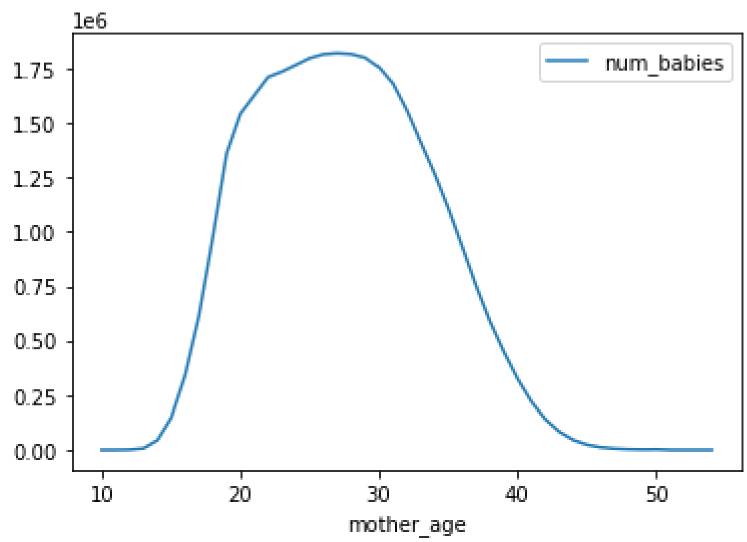
```
# Bar plot to see is_male with avg_wt linear and num_babies logarithmic
df = get_distinct_values('is_male')
df.plot(x='is_male', y='num_babies', kind='bar');
df.plot(x='is_male', y='avg_wt', kind='bar');
```

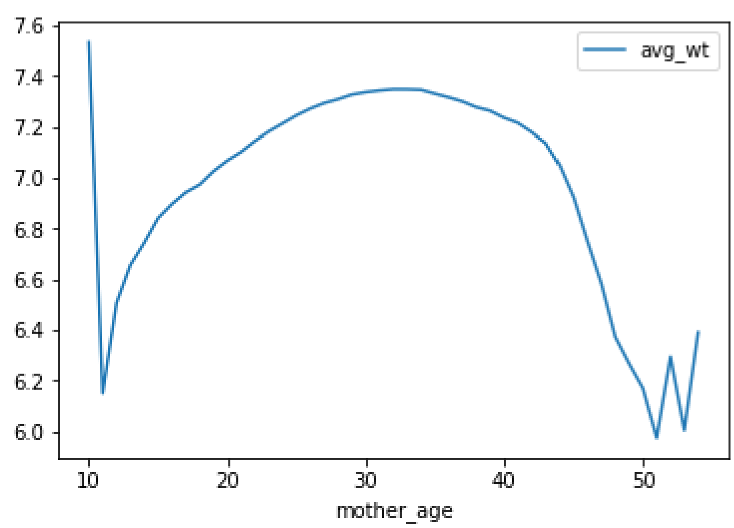




In [10]:

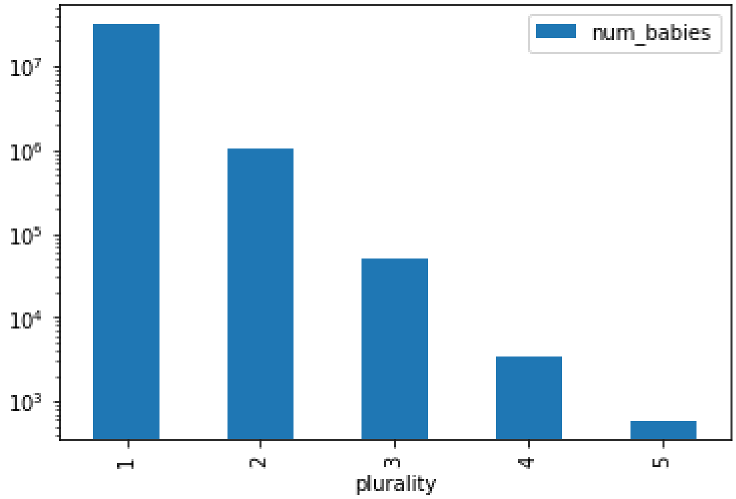
```
# Line plots to see mother_age with avg_wt linear and num_babies logarithmic
df = get_distinct_values('mother_age')
df = df.sort_values('mother_age')
df.plot(x='mother_age', y='num_babies');
df.plot(x='mother_age', y='avg_wt');
```

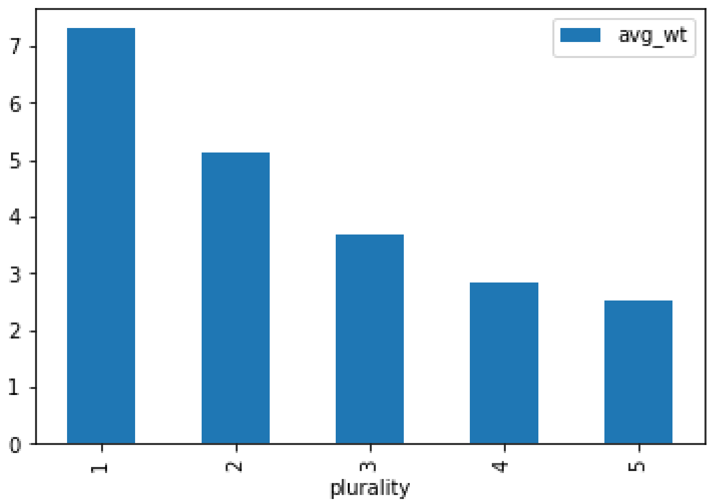




In [11]:

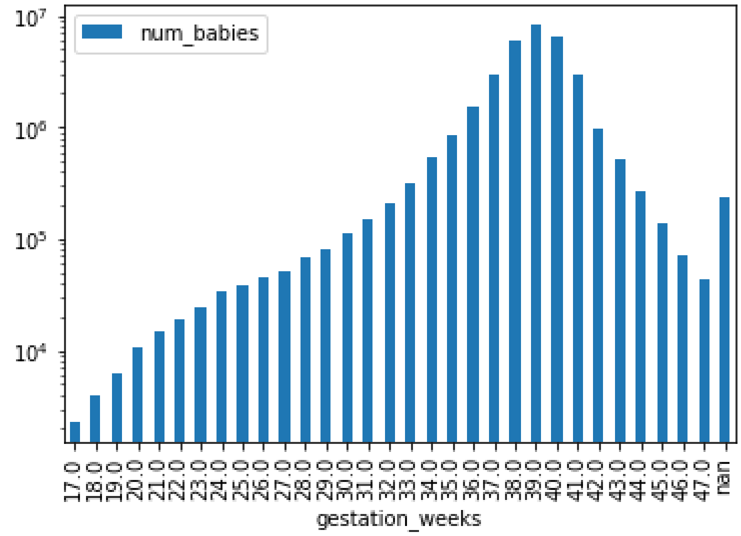
```
# Bar plot to see plurality(singleton, twins, etc.) with avg_wt linear and num_b
df = get_distinct_values('plurality')
df = df.sort_values('plurality')
df.plot(x='plurality', y='num_babies', logy=True, kind='bar');
df.plot(x='plurality', y='avg_wt', kind='bar');
```

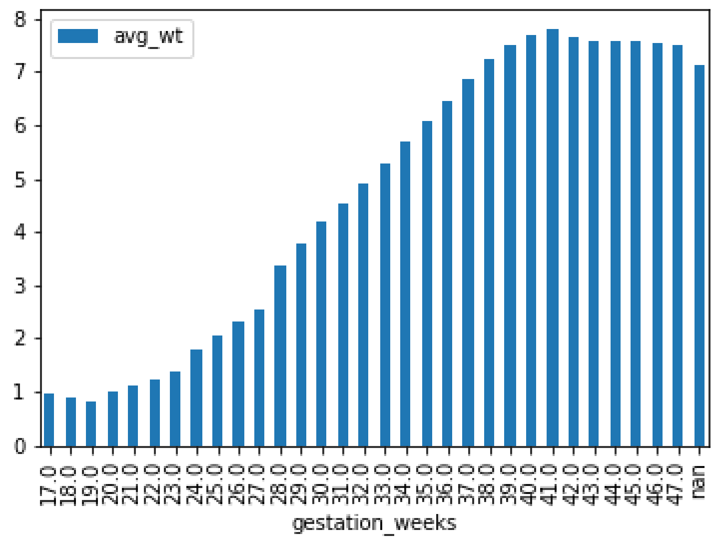




In [12]:

```
# Bar plot to see gestation_weeks with avg_wt linear and num_babies logarithmic
df = get_distinct_values('gestation_weeks')
df = df.sort_values('gestation_weeks')
df.plot(x='gestation_weeks', y='num_babies', logy=True, kind='bar');
df.plot(x='gestation_weeks', y='avg_wt', kind='bar');
```





All these factors seem to play a part in the baby's weight. Male babies are heavier on average than female babies. Teenaged and older moms tend to have lower-weight babies. Twins, triplets, etc. are lower weight than single births. Preemies weigh in lower as do babies born to single moms. In addition, it is important to check whether you have enough data (number of babies) for each input value. Otherwise, the model prediction against input values that doesn't have enough data may not be reliable.

In the next notebook, I will develop a machine learning model to combine all of these factors to come up with a prediction of a baby's weight.