探索数据集公开课讲义

<2018年 9月 14日更新 V1.0>

如果有更新迭代的建议，请发送邮件至 kylie@udacity.com 并抄送 april.chen@udacity.com。谢谢。

「课程重点」

* 数据分析过程 - 五个步骤

* 案例分析 - 五个步骤，NumPy, Pandas, Matplotlib, Sqlite 等数据分析库在案例中的运用

* SQL 知识

「公开课讲解知识点」

* 数据分析过程

* App Store 案例分析

  - 主要是以案例分析的方式，讲解数据分析过程、数据分析中常用的分析思路和方法、NumPy 和 Pandas 包以及常用函数的代码实现。
  - 得出结论的严谨性与局限性

* SQL 知识

0 使用说明

本讲义作为 Udacity 数据分析纳米学位课程的辅助，供 Udacity 数据分析入门 VIP 班级公开课使用，有对应的PPT，请助教结合PPT进行讲解。

本讲义内含三部分：

1. 数据分析过程 + Python
2. App Store 案例分析
3. SQL 知识

本课程+项目总时长为 4 周。讲义内的内容建议分 2-3 个公开课时讲解完毕，为同学必须要了解的知识内容，其余公开课助教可以根据同学的需求/常见问题自由控制主题。

公开课和讲义中包含 * 为拓展内容，不是必须讲解的部分。

建议时间安排：

第一课时：数据分析过程 + 案例分析（上）

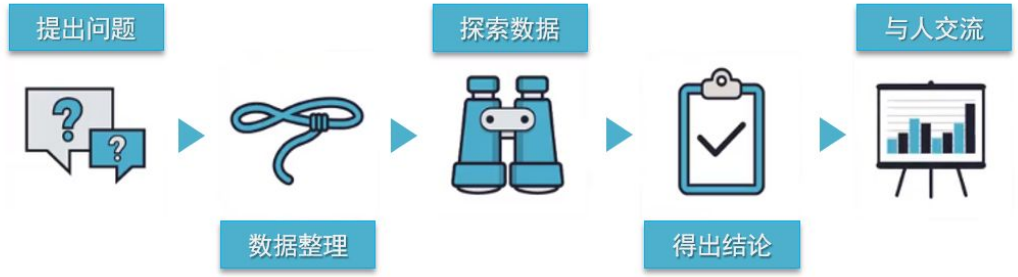
第二课时：案例分析（下）

第三课时：SQL 知识重点回顾

1 课程内容简介

数据分析入门的 P3 项目涉及到了两部分主要的内容：通过 Pandas 学习数据分析过程，以及对 SQL 知识的学习。我们的公开课也将分为两部分来讲解。

首先是关于数据分析过程的知识点。本课程以 Kaggle 上的 Mobile App Store (7200 apps) 数据集为例，讲解整个数据分析的过程。
数据分析过程分为 5 个部分：

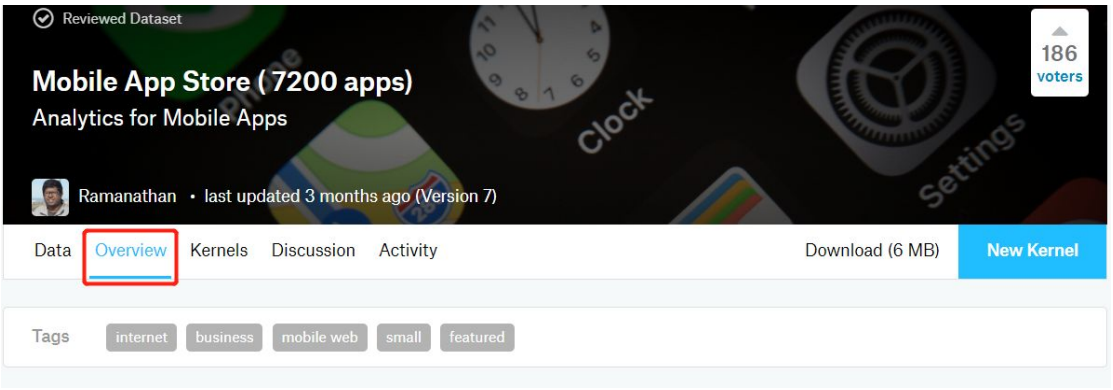


2 提出问题

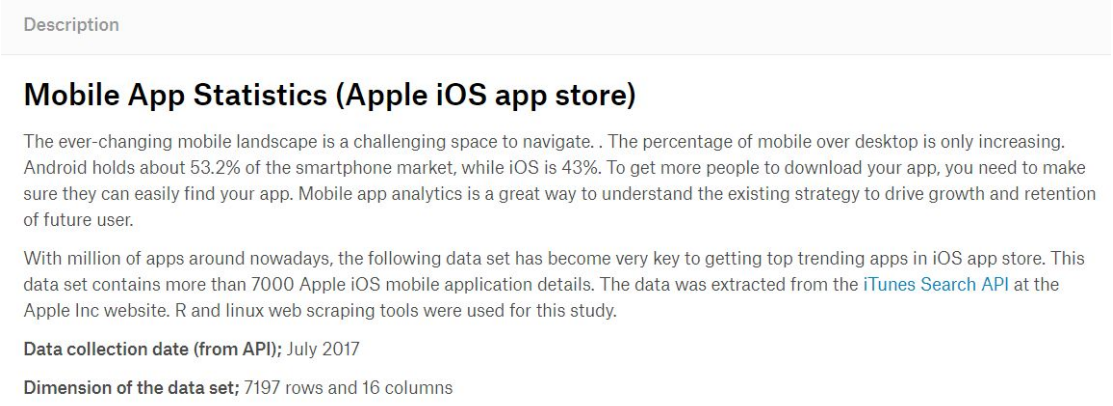
这里我们是使用现有数据集进行分析，而不是现有分析目标再去收集数据的过程。
（在进阶课程的数据清洗项目中，我们会详细学习各种收集数据的方法）

2.1 初步了解数据集

拿到一个数据集之后，我们需要了解数据集的背景知识。对于当前的数据集而言，我们可以去 Kaggle 网站这个数据集的页面去了解数据集的大致情况。



Kaggle 网站是比较好的数据来源，我们可以在每个数据集页面的 Overview 标签下了解数据集的含义：



数据的背景简介、描述以及每一列含义的数据字典，基本上都可以找到。

appleStore.csv

1. "id" : App ID
2. "track_name": App Name
3. "size_bytes": Size (in Bytes)
4. "currency": Currency Type
5. "price": Price amount
6. "rating_count_tot": User Rating counts (for all version)
7. "rating_count_ver": User Rating counts (for current version)
8. "user_rating" : Average User Rating value (for all version)
9. "user_rating_ver": Average User Rating value (for current version)
10. "ver" : Latest version code
11. "cont_rating": Content Rating
12. "prime_genre": Primary Genre
13. "sup_devices.num": Number of supporting devices
14. "ipadSc_urls.num": Number of screenshots showed for display
15. "lang.num": Number of supported languages
16. "vpp_lic": Vpp Device Based Licensing Enabled

appleStore_description.csv

1. id : App ID
2. track_name: Application name
3. size_bytes: Memory size (in Bytes)
4. app_desc: Application description

翻译如下：

appleStore.csv

1. "id" : App ID
2. “track_name”: App 名称
3. “size_bytes”: 大小 (以 Bytes 为单位)
4. “currency”: 价格货币单位
5. “price”: APP 价格
6. “rating_count_tot”: 用户评分数量 (所有版本的总计)
7. “rating_count_ver”: 用户评分数量 (当前版本)
8. “user_rating” : 用户平均评分 (所有版本)
9. “user_rating_ver”: 用户平均评分 (当前版本)
10. “ver” : 最新版本号
11. “cont_rating”: 内容评级
12. “prime_genre”: 主要类型
13. “sup_devices.num”: 支持设备数量
14. “ipadSc_urls.num”: 展示截图的数量

15. “lang.num”: 支持语言的数量
16. "vpp_lic": Vpp Device Based Licensing Enabled

appleStore_description.csv

1. id : App ID
2. track_name: App 名称
3. size_bytes: 大小 (以 Bytes 为单位)
4. app_desc: APP 描述文字

除了从数据集的简介中了解数据，最直观的方式还是观察数据本身。让我们现在打开 Notebook，开始数据探索之旅！
首先，导入所有需要的库，本项目所需要用到的库基本上就是 pandas、numpy、matplotlib，如果有其他绘图风格的需求，可能还会导入 seaborn 这样的绘图库，最后不要忘记 magic 命令。

```
In [1]: # 导入常用库
        import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
```

然后使用 pd.read_csv 导入数据集，使用 head、 tail或者 sample简单查看一下数据集的内容：

```
In [2]: # 读取数据集
        df_desc = pd.read_csv('appleStore_description.csv')
        df_app = pd.read_csv('AppleStore.csv')
```

```
In [3]: df_desc.head(3)
```

Out[3]:

	id	track_name	size_bytes	app_desc
0	281656475	PAC-MAN Premium	100788224	SAVE 20%, now only $3.99 for a limited time!\n...
1	281796108	Evernote - stay organized	158578688	Let Evernote change the way you organize your ...
2	281940292	WeatherBug - Local Weather, Radar, Maps, Alerts	100524032	Download the most popular free weather app pow...

```
In [4]: df_app.head(3)
```

Out[4]:

	Unnamed: 0	id	track_name	size_bytes	currency	price	rating_count_tot	rating_coun
0	1	281656475	PAC-MAN Premium	100788224	USD	3.99	21292	
1	2	281796108	Evernote - stay organized	158578688	USD	0.00	161065	
2	3	281940292	WeatherBug - Local Weather, Radar, Maps, Alerts	100524032	USD	0.00	188583	

2.2 提出感兴趣的问题

在初步了解数据集的含义，以及其中所包含的具体内容之后，我们可以有一个大概的思路或者直觉，提出自己感兴趣的问题，比如说：

- 这些 APP 的价格分布如何，免费应用所占比例是多少？
- 总评分与价格有没有相关性？
- APP的分类收费情况？
- 各个分类的评分分布差异？

2.3 问题迭代

随着对数据集了解的逐渐深入，我们可能会回过头再修改最开始提出的问题。

3 数据整理

数据整理的过程包括数据评估和数据清洗。

3.1 数据评估

Pandas 常用的评估方法有：head, sample, info, describe, isnull, unique, duplicated, value_counts 等：

查看数据内容	缺失值、重复项	异常值
df.head(n)	df.info()	df.describe()
df.sample(n)	df.isnull().sum()	df['column'].unique()
df.tail(n)	df.duplicated().sum()	df['column'].value_counts()

3.2 数据清洗

以下是一些常用的数据清理思路：

- 合并数据集
  如果有多个数据集，都是针对同一观察对象，那么需要合并为一个数据集。
- 针对提出问题筛选数据列
  这样可以减少一些数据清洗操作，筛选出所需特征列有两种方法：一种是提取出所需列，一种是删除不需要的列：
  - df = df[['column1','column2']]
  - df = df.drop(['column3','column4'],axis=1)
- 缺失值和重复项
  - df.dropna(subset=['column1','column2'], inplace = True)
  - df.fillna({'column1': 0, 'column2': 'No_value'})
  - df.drop_duplicates(subset=['column1','column2'], inplace = True)

- 类型修改
  - df['column'] = df['column'].astype(str)
  - df.dtypes
  - df['column'].dtype
- 异常值处理
  - replace
  - df.loc[mask,column]=value
  - df.drop(index)

Pandas 常用数据清洗方法列表：

方法示例	含义
pd.merge(df1, df2, on='id')	将 df1 与 df2 按照 id 列进行合并
pd.concat([df1, df2],axis=1)	将 df2 中的列添加到 df1 的尾部
df.drop(column=['c1', 'c2'])	删除 df 中的 c1 和 c2 列
df[['col1', 'col2']]	筛选出 df 中的 col1 和 col2
df.set_index('col1')	更改索引列为 col1
df.reset_index()	将 df 的索引重设为数字，原来的索引转变为普通列
df.sort_values('col')	按照 col 列进行排序，默认为从小到大的顺序
df.dropna(subset=['c1','c2'])	删除所有 c1, c2 列中有缺失值的行
df.fillna(value)	将 df 中的所有 NaN 替换为 value
df.fillna({'c1':0, 'c2':False})	按列替换 NaN 为不同的值
df.drop_duplicates()	删除重复项，也可以设定 subset
df['col'].astype(float)	将 col 列的类型转换为 float 类型
df.rename(columns= {'y':'year'})	修改列名
df.replace([1,3],['one','three'])	用 'one' 代替 1，用 'three' 代替 3
df.loc[df['age']<0, 'age'] = 0	选出 df 中所有 age 列小于 0 的行，将其 age 特征重新赋值

3.3 在APP数据集上应用上述方法

针对最开始提出的问题，以及对数据的更详细的评估，我们可以对数据进行以下处理：

- 将 df_desc 中的 app_desc 列合并到 df_app 中
- 删除不需要的数据列（'Unnamed: 0','currency','vpp_lic'）

- 将列名中的 . 修改为 _
- id 列修改为字符串类型
- 对 prize 进行 cut 分段，创建新的分类变量 price_cut

具体的代码如下所示：

```
df = pd.merge(df_app,df_desc[['id','app_desc']],on='id')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 7197 entries, 0 to 7196
Data columns (total 18 columns):
Unnamed: 0          7197 non-null int64
id                  7197 non-null int64
track_name          7197 non-null object
size_bytes          7197 non-null int64
currency            7197 non-null object
price               7197 non-null float64
rating_count_tot    7197 non-null int64
rating_count_ver    7197 non-null int64
user_rating         7197 non-null float64
user_rating_ver     7197 non-null float64
ver                 7197 non-null object
cont_rating         7197 non-null object
prime_genre         7197 non-null object
sup_devices.num     7197 non-null int64
ipadSc_urls.num     7197 non-null int64
lang.num            7197 non-null int64
vpp_lic             7197 non-null int64
app_desc            7197 non-null object
dtypes: float64(3), int64(9), object(6)
memory usage: 1.0+ MB
```

```
df.drop(['Unnamed: 0','currency','vpp_lic'],axis=1,inplace=True)
```

```
df = df.rename(columns={c: c.replace('.', '_') for c in df_app.columns})
df.columns
```

```
Index(['id', 'track_name', 'size_bytes', 'price', 'rating_count_tot',
       'rating_count_ver', 'user_rating', 'user_rating_ver', 'ver',
       'cont_rating', 'prime_genre', 'sup_devices_num', 'ipadSc_urls_num',
       'lang_num', 'app_desc'],
      dtype='object')
```

```
df.id = df.id.astype('str')
df.id.dtype
```

```
dtype('O')
```

```
price_labels = ['Free','USD:0~5','USD:5~20','USD:20+']
df['price_cut'] = pd.cut(df['price'],bins=[-1,0,5,20,300],labels=price_labels)
```

```
df['price_cut'].value_counts()
```

```
Free        4056
USD:0~5     2703
USD:5~20     402
USD:20+       36
Name: price_cut, dtype: int64
```

在数据清洗的过程中，每完成一个数据问题的清洗都要进行对应的检测。如果不检测的话，那么全部清洗完之后的结果与预期不符就需要再一一追溯，非常麻烦。

3.4 问题迭代

现在我们已经完成了数据整理的操作，对数据的了解更深入了，这个时候可以对最开始提出的问题进行迭代（对问题的迭代可能发生在任何一步中），可能会提出更有价值的问题。

比如在探索了 prime_genre 之后了解到游戏类应用占据了数据集的 53.7%，其他分类的类型很多，但是每个类别中数量较少，所以将关于分类的问题修改为游戏类与非游戏类的对比。

迭代后的问题：

- 游戏类与非游戏类 APP 的收费情况？
- 游戏类与非游戏类 APP 的评分分布差异？

迭代后所需要进一步清洗的内容：

```
df['is_Game'] = df['prime_genre'] == 'Games'
df['is_Game'].replace([True, False], ['Games','NotGames'], inplace=True)
```

```
df['is_Game'].value_counts()
```

```
Games       3862
NotGames    3335
Name: is_Game, dtype: int64
```

4 探索数据

针对想要探索的问题，对清洗后的数据集进行探索。在本项目中，不要求统计检验和机器学习建模的方法来探索。只需要对数据进行可视化分析以及一些简单的统计计算得出一些暂时性的结论即可。

4.1 简单统计计算方法

- df.describe()：查看数据值列的汇总统计
- df.mean()：返回所有列的均值，也可以直接对某一列使用
- df.count()：返回每一列中的非空值的个数，也可以直接对某一列使用
- df.max()：返回每一列的最大值，也可以直接对某一列使用
- df.min()：返回每一列的最小值，也可以直接对某一列使用
- df.median()：返回每一列的中位数，也可以直接对某一列使用
- df.std()：返回每一列的标准差，也可以直接对某一列使用
- df.corr()：返回列与列之间的相关系数
- df['col1'].corr(df['col2']): 返回 col1 列与 col2列 的相关系数

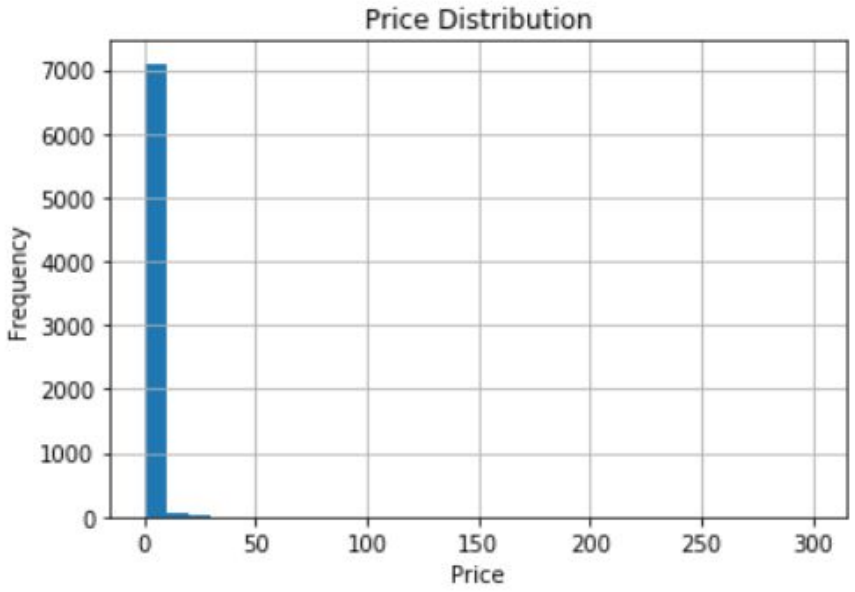
4.2 单变量探索

单变量探索，基本上在数据评估时就已经进行了一些探索。在探索数据阶段，我们可以使用一些可视化来得出更加明显的结论。

变量有连续型数值变量、离散型数值变量和分类变量。对于不同的变量使用不同的可视化种类：

- 连续型数值变量
  - 直方图

```
df['price'].hist(bins=30,figsize=(6,4))
plt.title('Price Distribution')
plt.ylabel('Frequency')
plt.xlabel('Price');
```



直方图可以修改 bins 参数控制图中bar 的数量，可以看出 price 是非常偏斜的状态。

- 箱线图

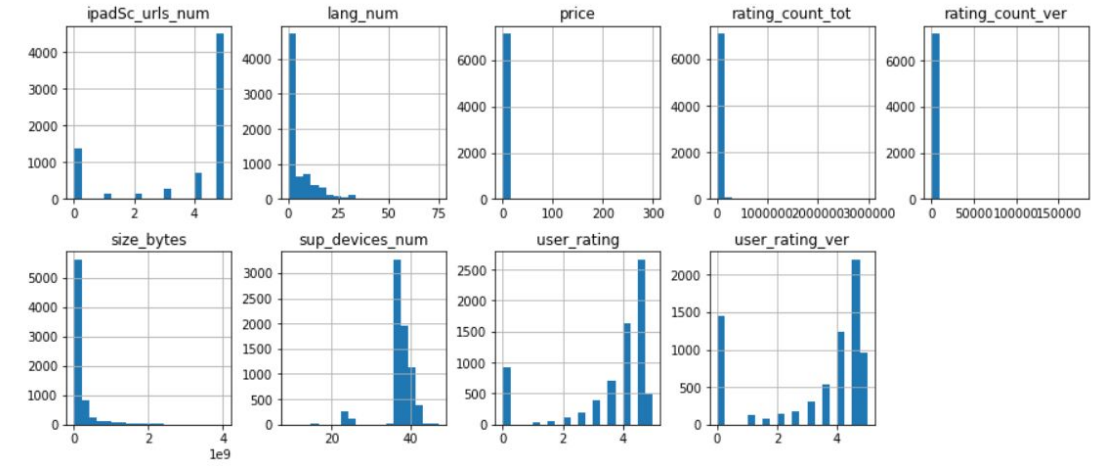
```
# showfliers=False 不显示离群值
df['price'].plot.box(showfliers=False, showmeans=True, figsize=(6,4))
plt.title('Price Distribution')
plt.ylabel('USD');
```

使用箱线图将离群值隐藏之后，可以发现大部分的价格都在 5 美元以下。

- 直方图矩阵

对数据集 df直接调用hist绘制直方图，可以看到df中所有数值数据的直方图矩阵：

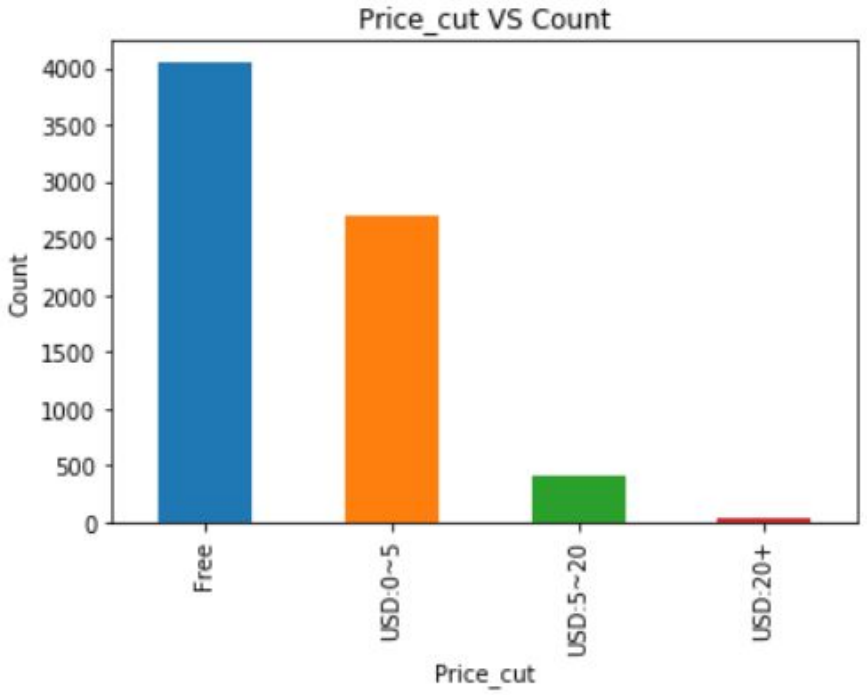
```
df.hist(figsize=(15,6),layout=(2,5),bins=20);
```



图中可以看到，有一些数值类型的数据是离散型的。

- 离散/分类变量
  - 柱状图

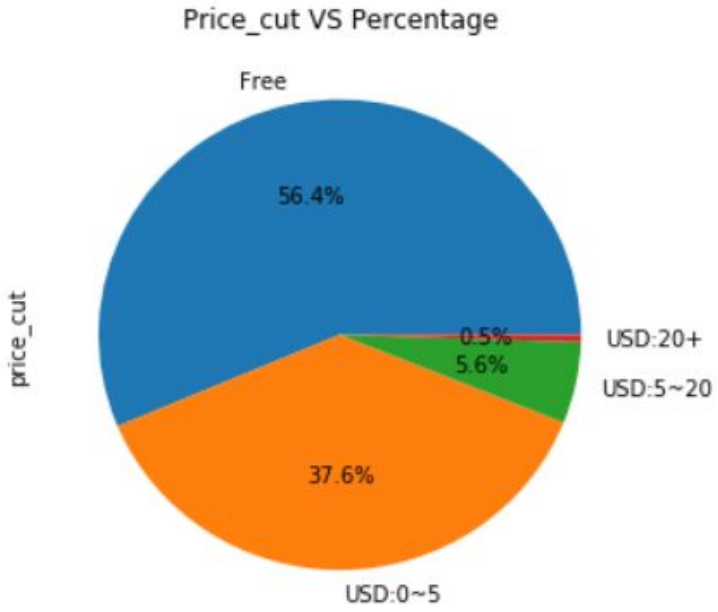
```
df.price_cut.value_counts().plot.bar(figsize=(6,4))
plt.title('Price_cut VS Count')
plt.ylabel('Count')
plt.xlabel('Price_cut');
```



图中可以明确看到分类变量各个类别的数量对比，在价格段：price_cut 这列数据中，占据绝大多数的类别是免费和5元以下。

  - 饼图

```
df.price_cut.value_counts().plot.pie(autopct='%.1f%%',figsize=(5,5))
plt.title('Price_cut VS Percentage');
```



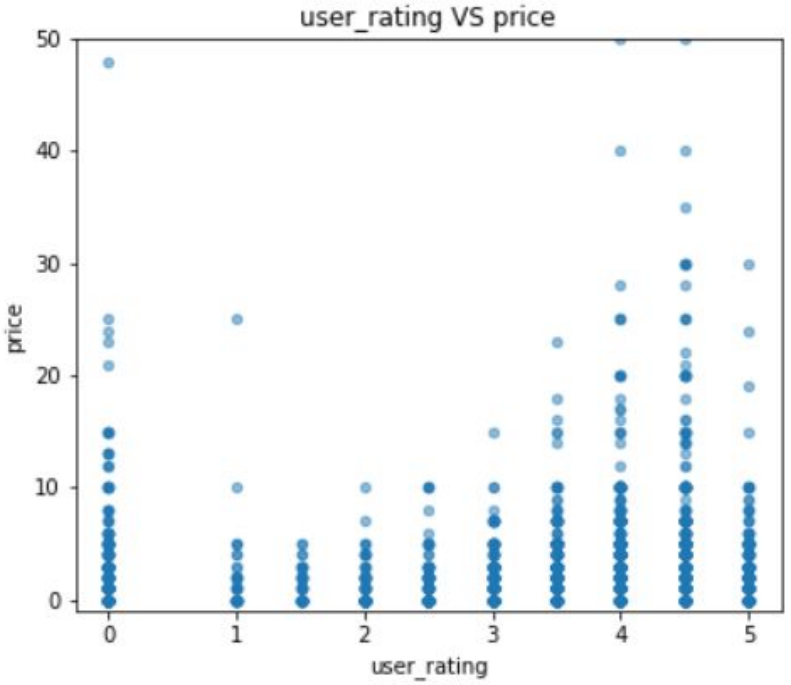
在饼图中可以看到各个类别所占比例，免费应用占据一半以上。

4.3 双变量探索

在双变量探索中，一般会探索两个变量之间是否存在相关性，或者某一变量随时间变量的变化趋势等。

- 两个数值变量的探索
  - 散点图
    散点图用于探索两个数值变量之间是否存在一些相关性。

```
df.plot.scatter(x='user_rating', y='price',alpha = 0.5,figsize=(6,5))
plt.title('user_rating VS price')
plt.ylim(-1,50);
```



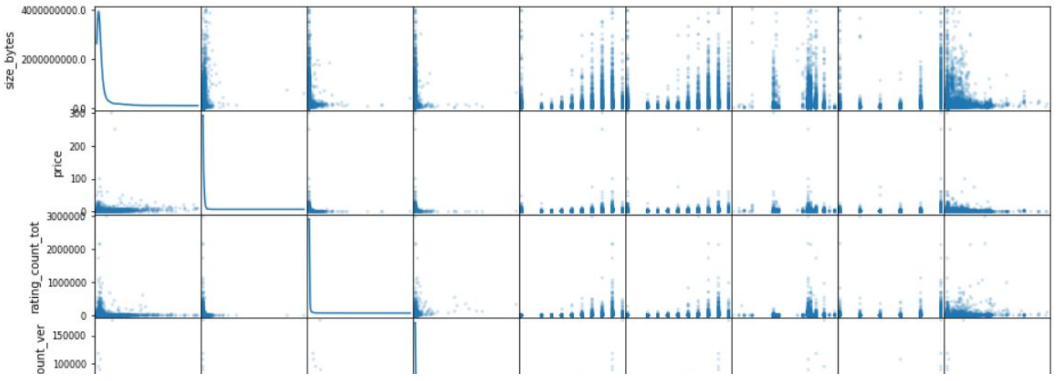
本次示例的数据集中，price 的分布存在一些极端值，所以使用 ylim 做了限制；因为 user_rating 实际上是离散数值，所以出现了一条一条的散点分布情况。

图中没有办法观察出明显的相关趋势

- 散点图矩阵

  对 df 中的所有数值变量可以绘制两两之间的散点图矩阵：

```
from pandas.plotting import scatter_matrix
scatter_matrix(df, alpha=0.2, figsize=(15, 15), diagonal='kde');
```



在探索相关性时，如果散点图给出的趋势不明显，还可以借助相关系数来获得更加明确的探索方向：

```
df.corr()
```

	size_bytes	price	rating_count_tot	rating_count_ver	user_rating	user_rating_ver	sup_devices_num	ipadSc_urls_num	lang_num
size_bytes	1.000000	0.182392	0.004486	0.006337	0.066256	0.086075	-0.118347	0.152697	0.004614
price	0.182392	1.000000	-0.039044	-0.018012	0.046601	0.025173	-0.115361	0.066100	-0.006713
rating_count_tot	0.004486	-0.039044	1.000000	0.163645	0.083310	0.088744	0.008832	0.015734	0.137675
rating_count_ver	0.006337	-0.018012	0.163645	1.000000	0.068754	0.077840	0.037951	0.024333	0.013287
user_rating	0.066256	0.046601	0.083310	0.068754	1.000000	0.774140	-0.042451	0.265671	0.170976
user_rating_ver	0.086075	0.025173	0.088744	0.077840	0.774140	1.000000	-0.018901	0.275737	0.175580
sup_devices_num	-0.118347	-0.115361	0.008832	0.037951	-0.042451	-0.018901	1.000000	-0.037728	-0.041681
ipadSc_urls_num	0.152697	0.066100	0.015734	0.024333	0.265671	0.275737	-0.037728	1.000000	0.088378
lang_num	0.004614	-0.006713	0.137675	0.013287	0.170976	0.175580	-0.041681	0.088378	1.000000

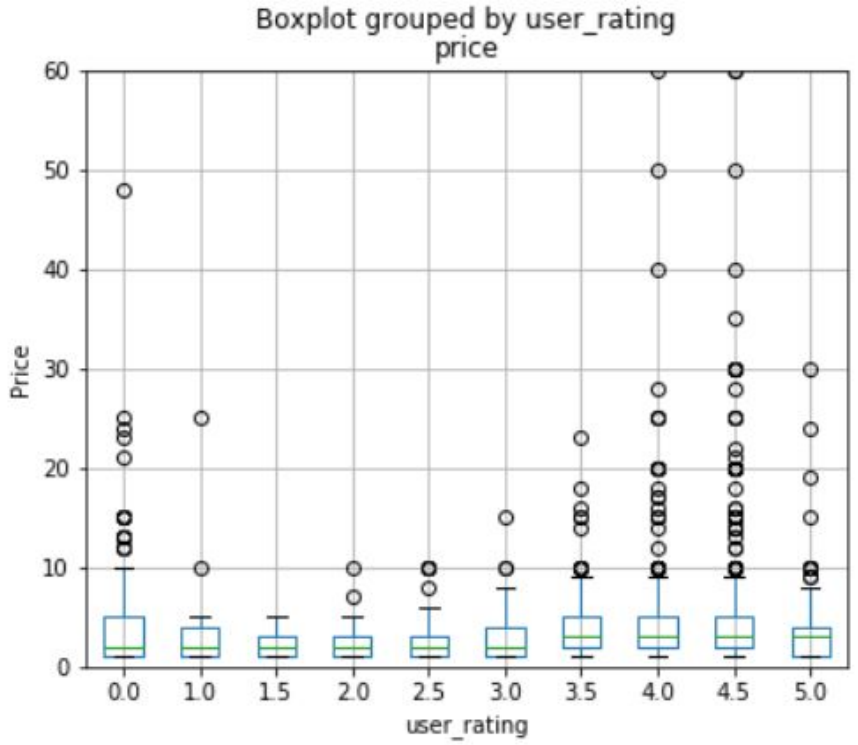
可以看到这些数值变量之间，只有 user_rating 和 user_rating_ver 之间的相关系数比较大，其他都没有超过0.3的情况。其他变量之间的线性相关性不大，我们使用其他可视化探索一下有没有其他数据规律。

- 连续-离散

  当两个变量的其中一个是连续数据，一个是离散或者分类数据，那么可以绘制箱线图或者对每个分类求均值、中位数、最大值最小值等进行柱状图对比：

  - 箱线图

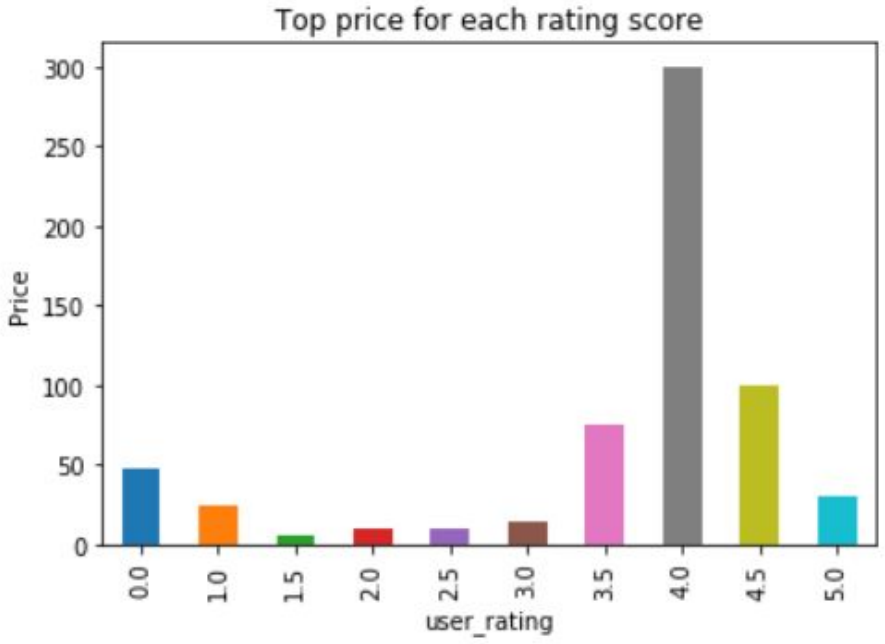
```
df[df['price']>0].boxplot(column=['price'],
                          by='user_rating',figsize=(6,5))
plt.ylim(0,60)
plt.ylabel('Price');
```



可以从图中看到，价格较高的一些离群值出现在3.5分以上的评分中比较多，0分的情况下，也比1~3分的情况下要多。看箱线区域，0分也与3.5~4.5的评分分布类似。这个现象比较有趣，猜测可能如果是价格较高的APP，如果让用户失望，可能会更倾向于评0分，而不是一些中间的分数。

  - 柱状图探索每个分类的均值、中位数、最大值最小值等

```
df.groupby('user_rating')['price'].max().plot.bar()
plt.title('Top price for each rating score')
plt.ylabel('Price');
```



每个评分的最大值探索，可以看到，最高价格的几个APP都分布在0分以及3.5分以上。与箱线图的结论相似。

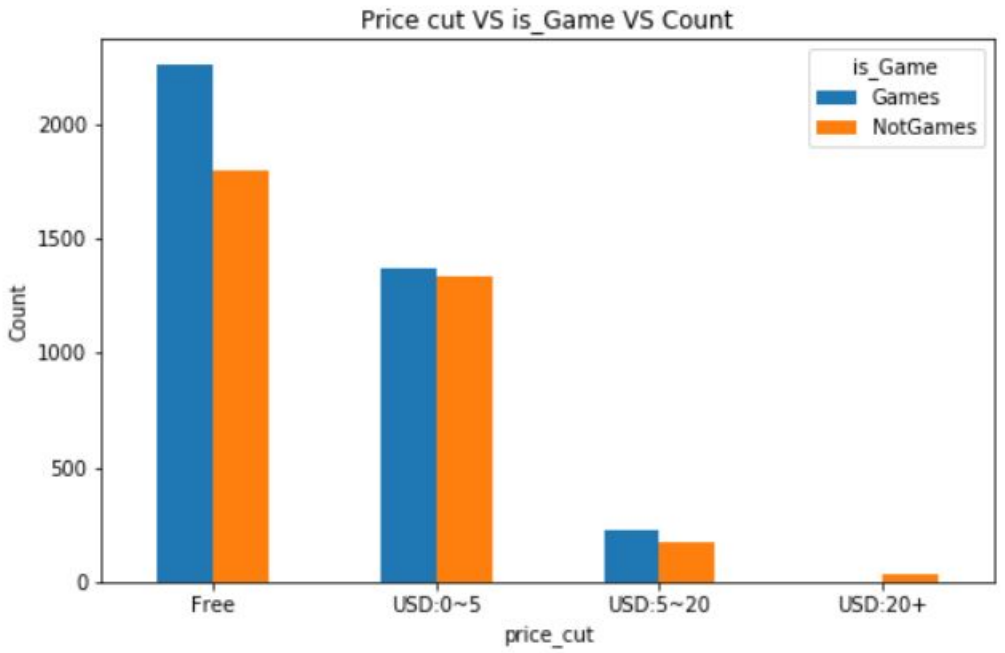
- 组合柱状图

  探索游戏应用与非游戏应用的价格差异，可以使用组合柱状图进行对比。下图代码中的 unstack 将数据由双索引的结构转换为宽格式的数据：

```
df.groupby(['price_cut','is_Game']).size().unstack()
```

is_Game	Games	NotGames
price_cut		
Free	2257	1799
USD:0~5	1372	1331
USD:5~20	230	172
USD:20+	3	33

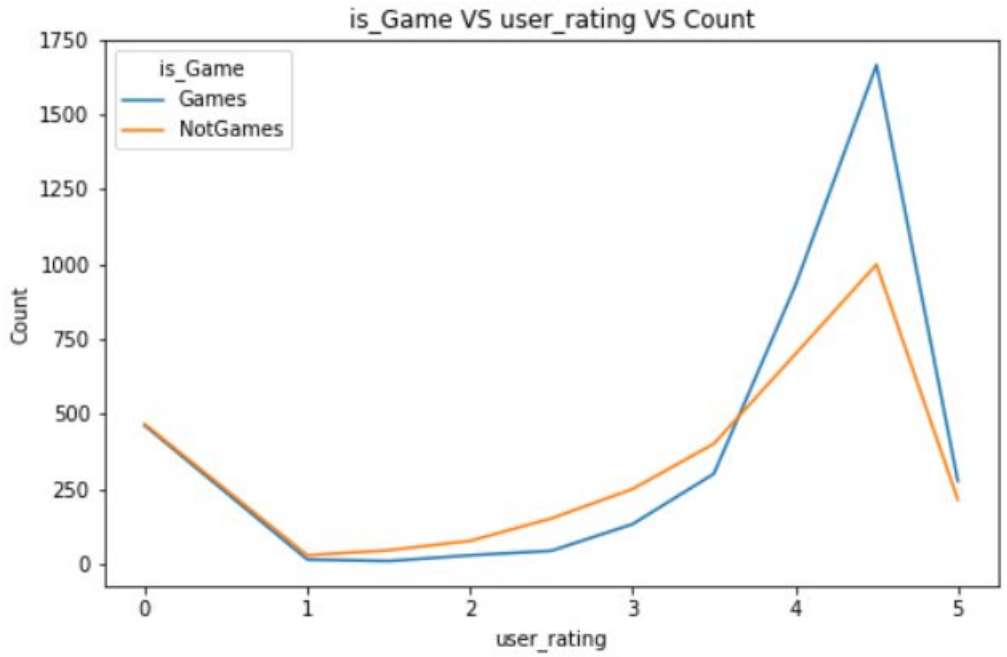
```
df.groupby(['price_cut','is_Game']).size().unstack().plot.bar(figsize=(8,5))
plt.title('Price cut VS is_Game VS Count')
plt.xticks(rotation=0) # 控制 x 轴刻度文字的角度
plt.ylabel('Count');
```



从图中可以看出，免费应用中游戏应用比例较高，非免费应用的分布较为接近。

- 折线图示例
  折线图主要用于绘制有前后关联的数据，最常见的是时间序列。本次数据集中没有包含时间序列的数据，因为 user_rating 的各个取值之间存在关联，所以尝试使用一下折线图绘制随着 user_rating 的增长，游戏与非游戏应用的数量变化：

```
df.groupby(['user_rating','is_Game']).size().unstack().plot(figsize=(8,5))
plt.title('is_Game VS user_rating VS Count')
plt.ylabel('Count');
```



图中可以看出，在3.5分之前，都是非游戏应用较多，3.5分之后则是游戏应用较多。

5 得出结论

5.1 总结

在数据探索的过程中，陆续得出了一些结论，在报告的最后部分，我们需要与最开始提出的问题相呼应，进行一一回答，也是对中间探索过程的一个总结。

5.2 局限性探讨

除了探索结果的总结，还需要对当前分析中存在的限制加以探讨。

- 数据本身的局限

是否存在一些你觉得可能对探索有帮助，但是数据集中不存在的特征？
是否有一些特征你不理解含义，或者不确定理解的是否正确？

- 探索方式的局限

  本次项目探索使用的只是一些简单的统计计算和可视化分析。这样得出的结论只能是暂时的，还有进一步验证的需要（使用统计检验或者机器学习建模等）。不过因为这个项目并不要求做其他工作，所以只需要声明这些局限即可。

5.3 当前分析报告的总结部分示例：

总结

- 所有 APP 中，大部分都小于 5 美元。免费应用占比为 56.4%，占据一半以上。
- 经过散点图探索和相关系数的计算，可以看到价格和评分没有明显的线性相关性。在深入探索后，可以发现价格较高的 APP 的评分比较极端，0分或者 3.5分以上。猜测可能如果是价格较高的 APP，如果让用户失望，可能会更倾向于评 0 分，而不是一些中间的分数。
- 免费应用中游戏应用比例较高，非免费应用中，游戏与非游戏类应用的比例则比较相似。
- 游戏/非游戏类应用与评分的关系：在 3.5分之前，都是非游戏应用较多，3.5分之后则是游戏应用较多。

局限性

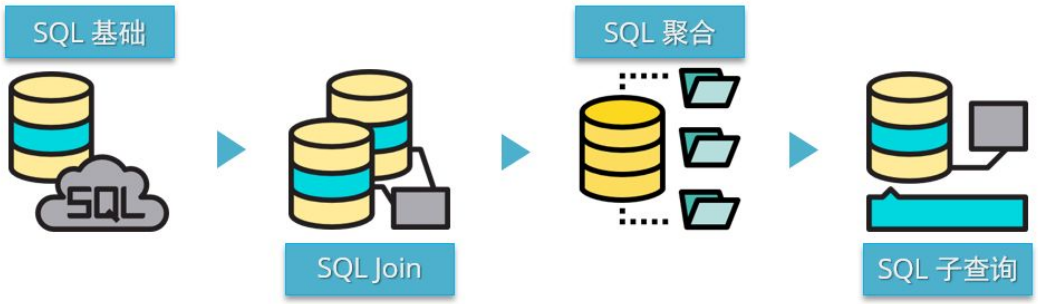
- 当前探索的问题主要集中在价格、评分、游戏分类这些特征，其他特征没有进行过多的探索。
- 根据对 APP 商店的了解，一些新上架的应用，评价应该会比较少，有可能有一些偶然情况不能了解到，评分结合评分人数应该会有更加准确的信息。
- 本次分析的过程只是使用了可视化探索和一些简单的统计计算，没有进行任何的显著性检验，所以目前的结论只是暂时的，还有进一步分析的可能性。

6 与人交流

以阅读者的角度，整理报告：

- 注意整理中间过程的迭代
  中间迭代的问题要整理记得添加到【简介：提出问题】部分；
- 数据整理过程添加思路的说明和所做变更的记录
  评估的目的和结果，相应的清洗方案
  做每个数据清洗的目的说明，清洗后对结果的说明
- 可视化图表的各种元素：
  可视化图表一定要记得添加标题、坐标轴标题、图例等要素。
  参考资料：matplotlib 绘图可视化知识点整理
- 每个可视化和统计计算后的推理说明文字
  推理说明要使用 Markdown 单元格。代码注释中只是针对代码功能的说明，而不是与分析思路和推理相关的说明。
  说明内容主要是：当前图表/统计计算表现了什么信息，从中我们可以推理出什么结果，是否可以回答所要探索的问题等。

7 SQL 知识点回顾



数据分析师使用 SQL 以查询为主，以下内容是对课程内容的知识点总结回顾。其中 SQL 查询语句的关键字用橙色英文大写标注。

SQL 基础

- SELECT：提供你想要的列
- FROM：提供列存在的表
- LIMIT：限制返回的行数
- ORDER BY：根据列对表排序，与DESC一起使用。
- WHERE：用于过滤结果的条件语句
- WHERE Col LIKE '%me% ：仅拉取文本中包含 'me' 的列
- WHERE Col IN ('Y','N')：仅过滤包含 'Y' 或 'N' 列的行
- WHERE Col NOT IN ('Y', 'N')：NOT经常与LIKE和IN一起使用
- WHERE Col1 > 5 AND Col2 < 3：过滤两个或多个条件必须为真的行
- WHERE Col1 > 5 OR Col2 < 3：过滤至少一个条件必须为真的行
- WHERE Col BETWEEN 3 AND 5：通常比使用AND的语法简单

SQL Join

- 主键：对于表格中的每行都是唯一的。主键通常是数据库中的第一列（就像Parch & Posey数据库中每个表格的id列）
- 外键：是出现在另一个表格中的主键，允许行不是唯一的行。
- JOIN：一种INNER JOIN，仅获取在两个表格中都存在的数据。
- LEFT JOIN：用于获取FROM中的表格中的所有行，即使它们不存在于JOIN语句中。
- RIGHT JOIN：用于获取JOIN中的表格中的所有行，即使它们不存在于FROM语句中。
- 其他高级JOIN：UNION和UNION ALL、CROSS JOIN、SELF JOIN
- 别名：使用AS或直接对表格和列设定别名。这样可以减少要输入的字符数，同时确保列标题可以描述表格中的数据。

SQL聚合

- NULL
  - NULL是一种数据类型，表示SQL中没有数据。在聚合中经常会忽略掉
- COUNT
  - 聚合数据的行数
  - COUNT不会考虑具有NULL值的行。因此，可以用来快速判断哪些行缺少数据。
- SUM
  - 对某一列聚合求和，你只能针对数字列使用SUM。
- MIN与MAX
  - MIN和MAX与COUNT相似，它们都可以用在非数字列上。
  - MIN将返回最小的数字、最早的日期或按字母表排序的最之前的非数字值，具体取决于列类型。
  - MAX则正好相反，返回的是最大的数字、最近的日期，或与“Z”最接近（按字母表顺序排列）的非数字值。
- AVG
  - AVG返回的是数据的平均值，即列中所有的值之和除以列中值的数量。
  - 该聚合函数同样会忽略分子和分母中的NULL值。
- GROUP BY
  - GROUP BY可以用来在数据子集中聚合数据。例如，不同客户、不同区域或不同销售代表分组。
  - SELECT语句中的任何一列，如果不在聚合函数中，则必须在GROUP BY条件中。
  - GROUP BY始终在WHERE和ORDER BY之间。
  - 你可以同时按照多列分组
- DISTINCT
  - 仅返回特定列的唯一值
  - 在使用DISTINCT时，尤其是在聚合函数中使用时，会让查询速度有所减慢。
- HAVING
  - HAVING是过滤被聚合的查询的“整洁”方式
  - 只要你想对通过聚合创建的查询中的元素执行WHERE条件，就需要使用HAVING
- DATE
  - DATE_TRUNC使你能够将日期截取到日期时间列的特定部分。
  - DATE_PART可以用来获取日期的特定部分
- CASE
  - CASE语句始终位于SELECT条件中
  - CASE必须包含以下几个部分：WHEN、THEN和END。ELSE是可选组成部分，用来包含不符合上述任一CASE条件的情况。
  - 你可以在WHEN和THEN之间使用任何条件运算符编写任何条件语句
  - 你可以再次包含多个WHEN语句以及ELSE语句，以便处理任何未处理的条件

SQL子查询

- 子查询
  - 子查询（SubQuery）或者说内查询（InnerQuery），也可以称作嵌套查询（NestedQuery）
  - 子查询用于为主查询返回其所需数据，或者对检索数据进行进一步的限制
  - 天气数据集示例：

```
SELECT c.year, c.avg_temp hangzhou, s.shanghai
FROM city_data c,
            (SELECT year, avg_temp shanghai
              FROM city_data WHERE city = 'Shanghai') AS s
WHERE c.city = 'Hangzhou'
AND c.year = s.year
```

修订记录

日期	版本	修改人	修改原因
18年 9月14日	V1.0 <定稿>	Kylie	补充教学目标、公开课讲解重点、0-使用说明
18年 9月9日	V0.1 <未定稿>	Ivy	初稿

感谢以下资深助教对本辅导资料的贡献

李伟伟 (Ivy)

审稿/修改负责人：Kylie（kylie@udacity.com）