五 分析函数

学习目标

- 掌握 LEAD，LAG，FIRST_VALUE，LAST_VALUE，NTH_VALUE 函数的使用方法

0 数据介绍

- 点击广告业务，通过用户点击广告收费，两张表格，一张访问信息统计表（STATISTICS），一张网站表（WEBSITE）
- 网站表（WEBSITE）：
  - `id`，`name` （网站名字），`budget` （每月预算）
  - `opened` 开始运营的日期

id	name	budget	opened
1	Gaming Heaven	3000	2016-02-01
2	All About Health	700	2016-03-15
3	Around The World	500	2016-05-01

- 访问信息统计表（STATISTICS）：
  - 此表记录了2016年5月的统计信息。每行数据均对应唯一的 `website_id` 和特定的日期 `day`
  - `user` ：显示当天该网站的UV（unique visit，独立IP，一个UV代表一个用户）
  - `impressions` ： 广告展示的次数
  - `clicks` ：是指广告的点击次数
  - `revenue` ：每日点击产生的收入

website_id	day	users	impressions	clicks	revenue
1	2016-05-01	36169	108507	237	66.34
1	2016-05-02	29580	295800	793	214.12
1	2016-05-03	30907	463605	1545	401.79
1	2016-05-04	19154	57462	160	38.31
1	2016-05-05	10897	163455	343	99.58
1	2016-05-06	24602	369030	804	184.92
1	2016-05-07	19882	139174	348	76.55
1	2016-05-08	26932	296252	782	117.25
1	2016-05-09	39275	117825	342	68.30
1	2016-05-10	28900	317900	1029	236.62
1	2016-05-11	23714	142284	423	84.69
1	2016-05-12	19006	171054	378	101.95
1	2016-05-13	24791	198328	526	89.43
1	2016-05-14	27617	165702	407	85.50
1	2016-05-15	8563	59941	135	33.75
1	2016-05-16	33679	303111	609	121.73
1	2016-05-17	25123	175861	383	57.47
1	2016-05-18	32233	225631	594	118.75
1	2016-05-19	33504	335040	857	197.08
1	2016-05-20	10830	86640	229	52.58
1	2016-05-21	13904	152944	380	75.90
1	2016-05-22	35180	386980	992	168.68
1	2016-05-23	18911	283665	773	154.59
1	2016-05-24	19938	259194	553	121.58

website_id	day	users	impressions	clicks	revenue
1	2016-05-25	14796	192348	416	66.61
1	2016-05-26	20953	146671	298	59.50
1	2016-05-27	14756	191828	564	84.63
1	2016-05-28	20397	203970	645	135.55
1	2016-05-29	30382	182292	446	71.31
1	2016-05-30	39977	519701	1382	262.61
1	2016-05-31	34817	382987	796	230.91
2	2016-05-01	7058	28232	106	30.78
2	2016-05-02	7716	46296	132	21.16
2	2016-05-03	6877	55016	144	34.66
2	2016-05-04	9498	47490	145	33.40
2	2016-05-05	8350	41750	128	38.54
2	2016-05-06	3508	28064	83	14.07
2	2016-05-07	5097	45873	202	60.63
2	2016-05-08	5491	10982	54	8.11
2	2016-05-09	3350	30150	78	17.23
2	2016-05-10	9669	48345	204	44.88
2	2016-05-11	8929	35716	149	29.76
2	2016-05-12	5758	17274	51	9.20
2	2016-05-13	6342	63420	202	48.47
2	2016-05-14	6219	49752	143	38.71
2	2016-05-15	5881	35286	126	36.42
2	2016-05-16	4959	19836	64	18.50
2	2016-05-17	9966	109626	359	100.64

website_id	day	users	impressions	clicks	revenue
2	2016-05-18	4182	41820	116	20.79
2	2016-05-19	3538	38918	193	40.46
2	2016-05-20	3584	17920	47	9.77
2	2016-05-21	5473	32838	124	28.50
2	2016-05-22	9484	66388	227	54.38
2	2016-05-23	5971	29855	119	30.93
2	2016-05-24	8085	32340	139	20.82
2	2016-05-25	3970	19850	78	20.32
2	2016-05-26	8805	79245	325	48.72
2	2016-05-27	9563	19126	70	10.51
2	2016-05-28	6682	80184	297	47.52
2	2016-05-29	6701	80412	228	36.45
2	2016-05-30	9558	105138	300	60.08
2	2016-05-31	5548	44384	178	53.26
3	2016-05-01	37	148	1	0.10
3	2016-05-02	73	292	1	0.21
3	2016-05-03	95	285	1	0.30
3	2016-05-04	32	224	1	0.15
3	2016-05-05	56	392	2	0.37
3	2016-05-06	100	1000	3	0.70
3	2016-05-07	167	668	2	0.39
3	2016-05-08	246	1722	8	1.59
3	2016-05-09	108	648	2	0.31
3	2016-05-10	158	1264	6	1.81

website_id	day	users	impressions	clicks	revenue
3	2016-05-11	216	2160	8	2.18
3	2016-05-12	187	1309	4	0.94
3	2016-05-13	254	1270	4	0.90
3	2016-05-14	107	535	3	0.60
3	2016-05-15	270	3240	9	1.65
3	2016-05-16	323	2584	11	2.41
3	2016-05-17	316	1264	4	0.83
3	2016-05-18	307	2763	8	1.81
3	2016-05-19	361	2527	11	3.07
3	2016-05-20	357	2856	9	2.54
3	2016-05-21	484	1452	7	1.09
3	2016-05-22	324	3888	12	2.58
3	2016-05-23	570	6840	19	4.24
3	2016-05-24	1664	8320	36	5.36
3	2016-05-25	2315	11575	30	5.10
3	2016-05-26	3586	28688	72	16.54
3	2016-05-27	1226	6130	20	3.00
3	2016-05-28	5998	29990	117	21.09
3	2016-05-29	7287	58296	166	39.86
3	2016-05-30	7785	23355	91	19.16
3	2016-05-31	1545	16995	55	9.96

1 LEAD(X)函数

- 与之前介绍的聚类函数和排序函数语法类似

```
<analytic function> OVER (...)
```

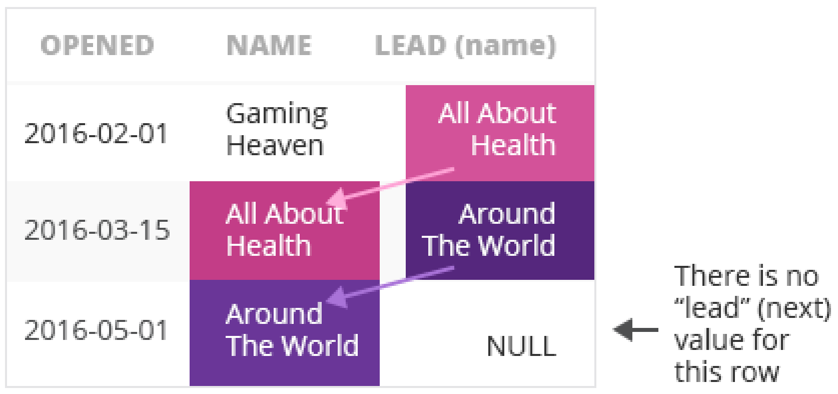
- 与聚类函数不同的地方是，分析函数只引用窗口中的单个行

LEAD(X)函数介绍

- 我们看下面的例子

```
SELECT
  name,
  opened,
  LEAD(name) OVER(ORDER BY opened)
FROM website;
```

- 上面的SQL中，分析函数为LEAD（name）。 LEAD中传入name列作为参数，将以 `ORDER BY` 排序后的顺序，返回当前行的下一行 `name` 列所对应的值，并在新列中显示，具体如下图所示：



OPENED	NAME	LEAD (name)
2016-02-01	Gaming Heaven	All About Health
2016-03-15	All About Health	Around The World
2016-05-01	Around The World	NULL

There is no "lead" (next) value for this row

- 注意：最后一列没有下一列结果所以这里显示NULL
- LEAD() 中传入的列名与排序的列可以不同

练习42

- 需求： 统计id 为1的网站，每天访问的人数以及下一天访问的人数
  - 返回字段： `day` 日期， `users` 访问人数， `lead` 下一天访问人数

```sql
SELECT
  day,
  users,
  LEAD(users) OVER(ORDER BY day) AS `lead`
FROM statistics
WHERE website_id = 1;
```

查询结果

day	users	lead
2016/5/1	36169	29580
2016/5/2	29580	30907
2016/5/3	30907	19154
2016/5/4	19154	10897
2016/5/5	10897	24602
2016/5/6	24602	19882
2016/5/7	19882	26932
2016/5/8	26932	39275
2016/5/9	39275	28900
2016/5/10	28900	23714
2016/5/11	23714	19006
2016/5/12	19006	24791
2016/5/13	24791	27617
2016/5/14	27617	8563
2016/5/15	8563	33679
2016/5/16	33679	25123
2016/5/17	25123	32233
2016/5/18	32233	33504
2016/5/19	33504	10830
2016/5/20	10830	13904
2016/5/21	13904	35180
2016/5/22	35180	18911
2016/5/23	18911	19938
2016/5/24	19938	14796

day	users	lead
2016/5/25	14796	20953
2016/5/26	20953	14756
2016/5/27	14756	20397
2016/5/28	20397	30382
2016/5/29	30382	39977
2016/5/30	39977	34817
2016/5/31	34817	NULL

使用LEAD()函数计算增量

- lead函数在计算增量的时候非常有用，比如我们想比较同一列两个值的差值

```
SELECT
  day,
  clicks,
  LEAD(clicks) OVER(ORDER BY day),
  clicks - LEAD(clicks) OVER(ORDER BY day)
FROM statistics
WHERE website_id = 2;
```

- 上面的查询计算了每日增量：最后一列显示了当日与次日之间的点击次数差异
  - 从业务角度来看，这可以很容易地告诉我们有关该网站的很多信息
    - 如果大多数增量是正的，且增量在逐渐变大，那么该网站业务可能处于上升期
    - 如果大多数是负的，那么需要找到收入下滑的原因

练习43

- 需求：统计id为1的网站，每日收入，后一天收入，以及每日收入的环比

```sql
SELECT
  day,
  revenue,
  LEAD(revenue) OVER(ORDER BY day) as `lead`,
  LEAD(revenue) OVER(ORDER BY day) - revenue as diff
FROM statistics
WHERE website_id = 1;
```

查询结果

day	revenue	lead	diff
2016/5/1	66.34	214.12	147.78
2016/5/2	214.12	401.79	187.67
2016/5/3	401.79	38.31	-363.48
2016/5/4	38.31	99.58	61.27
2016/5/5	99.58	184.92	85.34
2016/5/6	184.92	76.55	-108.37
2016/5/7	76.55	117.25	40.7
2016/5/8	117.25	68.3	-48.95
2016/5/9	68.3	236.62	168.32
2016/5/10	236.62	84.69	-151.93
2016/5/11	84.69	101.95	17.26
2016/5/12	101.95	89.43	-12.52
2016/5/13	89.43	85.5	-3.93
2016/5/14	85.5	33.75	-51.75
2016/5/15	33.75	121.73	87.98
2016/5/16	121.73	57.47	-64.26
2016/5/17	57.47	118.75	61.28
2016/5/18	118.75	197.08	78.33
2016/5/19	197.08	52.58	-144.5
2016/5/20	52.58	75.9	23.32
2016/5/21	75.9	168.68	92.78
2016/5/22	168.68	154.59	-14.09
2016/5/23	154.59	121.58	-33.01
2016/5/24	121.58	66.61	-54.97

day	revenue	lead	diff
2016/5/25	66.61	59.5	-7.11
2016/5/26	59.5	84.63	25.13
2016/5/27	84.63	135.55	50.92
2016/5/28	135.55	71.31	-64.24
2016/5/29	71.31	262.61	191.3
2016/5/30	262.61	230.91	-31.7
2016/5/31	230.91		

LEAD(x,y)

- LEAD函数还可以传入两个参数:
  - 参数1 跟传入一个参数时的情况一样：一列的列名
  - 参数2 代表了偏移量，如果传入2 就说明要以当前行为基准，向前移动两列作为返回值
- 举例:

```
SELECT
  name,
  opened,
  LEAD(opened,2) OVER(ORDER BY opened)
FROM website;
```

- 上面的SQL中，LEAD函数传入了2，当前行为第一行时，会返回第三行的值作为LEAD函数的结果

练习 44

- 需求：统计id为2的网站，在2016年5月1日到5月14日之间，每天的用户访问数量以及7天后的用户访问数量
- 需要注意，最后7行最后一列会返回NULL，因为最后7行没有7日后的数据

```sql
SELECT
  day,
  users,
  LEAD(users, 7) OVER(ORDER BY day) AS `lead`
FROM statistics
WHERE website_id = 2
  AND day BETWEEN '2016-05-01' AND '2016-05-14'
```

查询结果

day	users	lead
2016-05-01	7058	5491
2016-05-02	7716	3350
2016-05-03	6877	9669
2016-05-04	9498	8929
2016-05-05	8350	5758
2016-05-06	3508	6342
2016-05-07	5097	6219
2016-05-08	5491	null
2016-05-09	3350	null
2016-05-10	9669	null
2016-05-11	8929	null
2016-05-12	5758	null
2016-05-13	6342	null
2016-05-14	6219	null

LEAD(x,y,z)

- lead函数也可以接收三个参数，第三个参数用来传入默认值，应用场景是当使用lead函数返回null的时候，可以用第三个参数传入的默认值进行填充

- 练习44中，后7行出现了null,这里可以传入默认值，如-1，用来避免出现null的情况

```
SELECT
  day,
  users,
  LEAD(users, 7, -1) OVER(ORDER BY day) AS `lead`
FROM statistics
WHERE website_id = 2
  AND day BETWEEN '2016-05-01' AND '2016-05-14';
```

查询结果

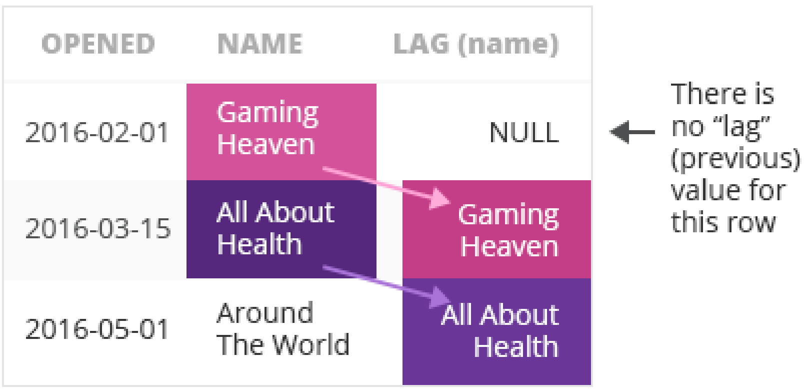
day	users	lead
2016-05-01	7058	5491
2016-05-02	7716	3350
2016-05-03	6877	9669
2016-05-04	9498	8929
2016-05-05	8350	5758
2016-05-06	3508	6342
2016-05-07	5097	6219
2016-05-08	5491	-1
2016-05-09	3350	-1
2016-05-10	9669	-1
2016-05-11	8929	-1
2016-05-12	5758	-1
2016-05-13	6342	-1
2016-05-14	6219	-1

2 LAG(x)函数

- LAG(x)函数与LEAD(x)用法类似，区别是，LEAD返回当前行后面的值，LAG返回当前行之前的值
- 示例：

  ```sql
  SELECT
    name,
    opened,
    LAG(name) OVER(ORDER BY opened)
  FROM website;
  ```

  - 上面的SQL会返回当前日期的前一行开业日期



OPENED	NAME	LAG (name)
2016-02-01	Gaming Heaven	NULL
2016-03-15	All About Health	Gaming Heaven
2016-05-01	Around The World	All About Health

There is no "lag" (previous) value for this row

- 注意：`LEAD(...)` 和 `LAG(...)` ,之间可以互相替换，可以在ORDER BY的时候通过 `DESC` 来改变排序方式，使 `LEAD(...)` 和 `LAG(...)` 返回相同结果，比如：

  ```sql
  LEAD (...) OVER(ORDER BY ...)
  ```

  与下面的写法相似

  ```sql
  LAG (...) OVER (ORDER BY ... DESC)
  ```

  再看：

  ```sql
  LEAD (...) OVER(ORDER BY ... DESC)
  ```

  与下面的写法相似

  ```sql
  LAG (...) OVER (ORDER BY ...)
  ```

练习45

- 需求：统计id为3的网站每天的点击数量，前一天的点击数量

```sql
SELECT
  day,
  clicks,
  LAG(clicks) OVER(ORDER BY day) as `lag`
FROM statistics
WHERE website_id = 3;
```

查询结果

day	clicks	lag
2016-05-01	1	null
2016-05-02	1	1
2016-05-03	1	1
2016-05-04	1	1
2016-05-05	2	1
2016-05-06	3	2
2016-05-07	2	3
2016-05-08	8	2
2016-05-09	2	8
2016-05-10	6	2
2016-05-11	8	6
2016-05-12	4	8
2016-05-13	4	4
2016-05-14	3	4
2016-05-15	9	3
2016-05-16	11	9
2016-05-17	4	11
2016-05-18	8	4
2016-05-19	11	8
……	……	……

LAG(x,y)

- 与LEAD(x,y)类似，LAG(x,y)返回当前行的前y行结果

```
SELECT
  name,
  opened,
  LAG(opened,2) OVER(ORDER BY opened)
FROM website;
```

- 使用LAG y = 2 ，所以返回的是两行以前的数据
- LEAD 和 LAG 容易记混
  - LEAD 领先的意思 找行号更大的数据
  - LAG 落后的意思 找行号更小的数据

练习46

- 需求：统计id = 3的网站每日广告收入以及三天前的广告收入

```
SELECT
  day,
  revenue,
  LAG(revenue, 3) OVER(ORDER BY day) AS `lag`
FROM statistics
WHERE website_id = 3;
```

查询结果

day	revenue	lag
2016-05-01	0.10	null
2016-05-02	0.21	null
2016-05-03	0.30	null
2016-05-04	0.15	0.10
2016-05-05	0.37	0.21
2016-05-06	0.70	0.30
2016-05-07	0.39	0.15
2016-05-08	1.59	0.37
2016-05-09	0.31	0.70
2016-05-10	1.81	0.39
2016-05-11	2.18	1.59
2016-05-12	0.94	0.31
2016-05-13	0.90	1.81
2016-05-14	0.60	2.18
2016-05-15	1.65	0.94
2016-05-16	2.41	0.90
2016-05-17	0.83	0.60
2016-05-18	1.81	1.65
2016-05-19	3.07	2.41
……	……	……

LAG(x,y,z)

- 与LEAD(x,y,z)一样，LAG(x,y,z) 最后一个参数是默认值，用来填补NULL值
- 修改前面的SQL，当LAG返回NULL时用-1填补

```
SELECT
  day,
  revenue,
  LAG(revenue, 3, -1.00) OVER(ORDER BY day)
FROM statistics
WHERE website_id = 3;
```

查询结果

day	revenue	lag
2016-05-01	0.10	-1.00
2016-05-02	0.21	-1.00
2016-05-03	0.30	-1.00
2016-05-04	0.15	0.10
2016-05-05	0.37	0.21
2016-05-06	0.70	0.30
……	……	……

练习47

- 每千次展示收入**RPM** (revenue per thousand impressions) 定义： 收入（`revenue`） **除以**展示次数（`impressions`） **乘 1000**.

  ```
  RPM = (revenue / impressions) * 1000
  ```

  For each statistics row with `website_id = 2`, show the `day`, the **RPM** and the **RPM 7 days later**. Rename the columns to `RPM` and `RPM_7`.
- 需求：统计id为2的网站，每天的RPM以及7日后的RPM
  - 返回字段： `day`, `RPM` 和 `RPM_7`

```sql
SELECT
  day,
  revenue / impressions * 1000 AS RPM,
  LEAD(revenue, 7) OVER(ORDER BY day) / LEAD(impressions, 7)
OVER(ORDER BY day) * 1000 AS RPM_7
FROM statistics
WHERE website_id = 2;
```

查询结果

day	RPM	RPM_7
2016/5/1	1.090252	0.738481
2016/5/2	0.457058	0.571475
2016/5/3	0.629998	0.928327
2016/5/4	0.703305	0.83324
2016/5/5	0.923113	0.532592
2016/5/6	0.501354	0.764269
2016/5/7	1.321692	0.778059
2016/5/8	0.738481	1.032137
2016/5/9	0.571475	0.932647
2016/5/10	0.928327	0.91803
2016/5/11	0.83324	0.49713
2016/5/12	0.532592	1.039621
2016/5/13	0.764269	0.5452
2016/5/14	0.778059	0.867896
2016/5/15	1.032137	0.819123
2016/5/16	0.932647	1.036007
2016/5/17	0.91803	0.643784
2016/5/18	0.49713	1.023677
2016/5/19	1.039621	0.614802
2016/5/20	0.5452	0.549513
2016/5/21	0.867896	0.592636
2016/5/22	0.819123	0.45329
2016/5/23	1.036007	0.571439
2016/5/24	0.643784	1.199981

day	RPM	RPM_7
2016/5/25	1.023677	NULL
2016/5/26	0.614802	NULL
2016/5/27	0.549513	NULL
2016/5/28	0.592636	NULL
2016/5/29	0.45329	NULL
2016/5/30	0.571439	NULL
2016/5/31	1.199981	NULL

练习48

- 转化率定义：转化率= 点击次数 `clicks` /展示次数 `impressions` *100
- 需求：统计id = 1的网站，5月15日至5月31日，每天点击次数 `clicks` ，展示次数 `impressions` ，转化率（ `conversion` ）和前一天的转化率（ `previous_conversion` ）

```
SELECT
  day,
  clicks,
  impressions,
  clicks / impressions * 100 AS conversion,
  LAG(clicks) OVER(ORDER BY day) / LAG(impressions) OVER(ORDER
BY day) * 100 AS previous_conversion
FROM statistics
WHERE website_id = 1
  AND day BETWEEN '2016-05-15' AND '2016-05-31';
```

查询结果

day	clicks	impressions	conversion	previous_conversion
2016-05-15	135	59941	0.22522146777664703600	null
2016-05-16	609	303111	0.20091649593713194200	0.22522146777664703600
2016-05-17	383	175861	0.21778563752054179200	0.20091649593713194200
2016-05-18	594	225631	0.26326169719586404400	0.21778563752054179200
2016-05-19	857	335040	0.25579035339063992400	0.26326169719586404400
2016-05-20	229	86640	0.26431209602954755300	0.25579035339063992400
2016-05-21	380	152944	0.24845695156397112700	0.26431209602954755300
2016-05-22	992	386980	0.25634399710579358100	0.24845695156397112700
2016-05-23	773	283665	0.27250453880457582000	0.25634399710579358100
2016-05-24	553	259194	0.21335370417525097000	0.27250453880457582000
2016-05-25	416	192348	0.21627466882941335500	0.21335370417525097000
2016-05-26	298	146671	0.20317581526000368200	0.21627466882941335500
2016-05-27	564	191828	0.29401338699251412700	0.20317581526000368200
2016-05-28	645	203970	0.31622297396675981800	0.29401338699251412700
2016-05-29	446	182292	0.24466240976016501000	0.31622297396675981800

day	clicks	impressions	conversion	previous_conversion
2016-05-30	1382	519701	0.26592213599742929100	0.24466240976016501000
2016-05-31	796	382987	0.20783995279213132600	0.26592213599742929100

3 FIRST_VALUE(x)函数

- FISRT_VALUE函数，从名字中能看出，返回指定列的第一个值

```
SELECT
  name,
  opened,
  budget,
  FIRST_VALUE(budget) OVER(ORDER BY opened)
FROM website;
```

- 上面的SQL中，我们按`opened`列进行排序，`FIRST_VALUE(budget)`返回的是开业最早的网站的预算`budget`

练习49

- 需求：统计id为2的网站每天用户访问情况，以及最少用户访问人数

```
SELECT
  day,
  users,
  FIRST_VALUE(users) OVER(ORDER BY users) as `first_value`
FROM statistics
WHERE website_id = 2;
```

查询结果

day	users	first_value
2016-05-09	3350	3350
2016-05-06	3508	3350
2016-05-19	3538	3350
2016-05-20	3584	3350
2016-05-25	3970	3350
2016-05-18	4182	3350
2016-05-16	4959	3350
2016-05-07	5097	3350
2016-05-21	5473	3350
2016-05-08	5491	3350
2016-05-31	5548	3350
2016-05-12	5758	3350
2016-05-15	5881	3350
2016-05-23	5971	3350
2016-05-14	6219	3350
2016-05-13	6342	3350
2016-05-28	6682	3350
2016-05-29	6701	3350
2016-05-03	6877	3350
2016-05-01	7058	3350

练习50

- 需求：统计id = 3 的网站收入情况，返回日期，收入，和第一天的收入

```sql
SELECT
  day,
  revenue,
  FIRST_VALUE(revenue) OVER(ORDER BY day) AS `first_value`
FROM statistics
WHERE website_id = 3;
```

查询结果

day	revenue	first_value
2016-05-01	0.10	0.10
2016-05-02	0.21	0.10
2016-05-03	0.30	0.10
2016-05-04	0.15	0.10
2016-05-05	0.37	0.10
2016-05-06	0.70	0.10
2016-05-07	0.39	0.10
2016-05-08	1.59	0.10
2016-05-09	0.31	0.10
2016-05-10	1.81	0.10
2016-05-11	2.18	0.10
2016-05-12	0.94	0.10
2016-05-13	0.90	0.10
2016-05-14	0.60	0.10
2016-05-15	1.65	0.10
2016-05-16	2.41	0.10
2016-05-17	0.83	0.10
2016-05-18	1.81	0.10
2016-05-19	3.07	0.10
……	……	……

4 LAST_VALUE(x)函数

- FIRST_VALUE(x)返回第一个值，LAST_VALUE(x)返回最后一个值

```sql
SELECT
  name,
  opened,
  LAST_VALUE(opened) OVER(ORDER BY opened)
FROM website;
```

- LAST_VALUE(opened) 返回最近开始营业的网站，我们运行一下

name	opened	last_value
Gaming Heaven	2016-02-01	2016-02-01
All About Health	2016-03-15	2016-03-15
Around The World	2016-05-01	2016-05-01

- 查询结果与我们预期的有些出入，它只返回了当前行的结果，而不是我们想要的最后一个值

LAST_VALUE 与 window frame

- 在上面的例子中，我们没有得到想要的结果，回顾一下之前我们所介绍的 window frame
  - 当 `OVER` 子句中包含 `ORDER BY` 时，如果我们不显式定义window frame，SQL会自动带上默认的window frame语句：

    `RANGE UNBOUNDED PRECEDING`，意味着我们的查询范围被限定在第一行到当前行（`current row`）
  - 如果想通过LAST_VALUE 与ORDER BY配合得到所有数据排序后的最后一个值，需要吧window frame语句写成
    - `RANGE BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING` 或者
    - `ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING`
- 修改上面的SQL

```sql
SELECT
  name,
  opened,
  LAST_VALUE(opened) OVER(
    ORDER BY opened
    RANGE BETWEEN UNBOUNDED PRECEDING
      AND UNBOUNDED FOLLOWING) AS `last_value`
FROM website;
```

查询结果

name	opened	last_value
Gaming Heaven	2016-02-01	2016-05-01
All About Health	2016-03-15	2016-05-01
Around The World	2016-05-01	2016-05-01

- 从上面的结果中可以看出，调整了window frame之后我们可以得到 `LAST_VALUE` 想要的结果
- `FIRST_VALUE` 使用默认的window frame就可以正常工作，但是 `LAST_VALUE` 想要得到预期的结果需要手动修改window frame
- 与 `FISRT_VALUE` 类似，我们在使用 `LAST_VALUE` 时，传入的字段与排序的字段可以有区别

练习51

- 需求：统计id为1的网站的广告展示情况，返回每日日期，广告展示次数，以及访问用户最多的一天广告展示的次数

```sql
SELECT
  day,
  impressions,
  LAST_VALUE(impressions) OVER(
    ORDER BY users
    ROWS BETWEEN UNBOUNDED PRECEDING
      AND UNBOUNDED FOLLOWING) AS `last_value`
FROM statistics
WHERE website_id = 1;
```

查询结果

day	impressions	last_value
2016-05-15	59941	519701
2016-05-20	86640	519701
2016-05-05	163455	519701
2016-05-21	152944	519701
2016-05-27	191828	519701
2016-05-25	192348	519701
2016-05-23	283665	519701
2016-05-12	171054	519701
2016-05-04	57462	519701
2016-05-07	139174	519701
2016-05-24	259194	519701
2016-05-28	203970	519701
2016-05-26	146671	519701
2016-05-11	142284	519701
2016-05-06	369030	519701
2016-05-13	198328	519701
2016-05-17	175861	519701
2016-05-08	296252	519701
2016-05-14	165702	519701
……	……	……

练习52

- 需求：统计id为1的网站，每日的访问用户数，最后一天的访问用户数，每日用户数与最后一天用户数的差值

```
SELECT
  day,
  users,
  LAST_VALUE(users) OVER(
    ORDER BY day
    ROWS BETWEEN UNBOUNDED PRECEDING
      AND UNBOUNDED FOLLOWING) `last_day_users`,
  users - LAST_VALUE(users) OVER(
    ORDER BY day
    ROWS BETWEEN UNBOUNDED PRECEDING
      AND UNBOUNDED FOLLOWING) `diff`
FROM statistics
WHERE website_id = 1;
```

查询结果

day	users	last_day_users	diff
2016-05-01	36169	34817	1352
2016-05-02	29580	34817	-5237
2016-05-03	30907	34817	-3910
2016-05-04	19154	34817	-15663
2016-05-05	10897	34817	-23920
2016-05-06	24602	34817	-10215
2016-05-07	19882	34817	-14935
2016-05-08	26932	34817	-7885
2016-05-09	39275	34817	4458
2016-05-10	28900	34817	-5917
2016-05-11	23714	34817	-11103
2016-05-12	19006	34817	-15811
2016-05-13	24791	34817	-10026
2016-05-14	27617	34817	-7200
2016-05-15	8563	34817	-26254
2016-05-16	33679	34817	-1138
2016-05-17	25123	34817	-9694
2016-05-18	32233	34817	-2584
2016-05-19	33504	34817	-1313
……	……	……	……

5 NTH_VALUE(x,n)函数

- 本小节最后一部分要介绍的就是 `NTH_VALUE(x,n)` 函数
- `NTH_VALUE(x,n)` 函数返回 x列，按指定顺序的第n个值

```
SELECT
  name,
  opened,
  NTH_VALUE(opened, 2) OVER(
    ORDER BY opened
    ROWS BETWEEN UNBOUNDED PRECEDING
      AND UNBOUNDED FOLLOWING)
FROM website;
```

- 上面的SQL将数据按照开业日期排序， `NTH_VALUE(opened,2)` 返回开业日期排在第二位的值
- 需要注意，我们需要调整window frame 否则某些情况下不能返回正确的数据
- 提示：可以在排序的时候加上 `DESC` 调整排序的顺序，配合 `NTH_VALUE(x,n)` 在某些场景下更加方便

练习 53

- 需求：统计id为2的网站的收入情况，在5月15和5月31日之间，每天的收入，以及这半个月内的第三高的日收入金额

```
SELECT
  day,
  revenue,
  NTH_VALUE(revenue,3) OVER (
    ORDER BY revenue DESC
    ROWS BETWEEN UNBOUNDED PRECEDING
      AND UNBOUNDED FOLLOWING) `3rd_highest`
FROM statistics
WHERE website_id = 2
  AND day BETWEEN '2016-05-15' AND '2016-05-31';
```

查询结果

day	revenue	3rd_highest
2016-05-17	100.64	54.38
2016-05-30	60.08	54.38
2016-05-22	54.38	54.38
2016-05-31	53.26	54.38
2016-05-26	48.72	54.38
2016-05-28	47.52	54.38
2016-05-19	40.46	54.38
2016-05-29	36.45	54.38
2016-05-15	36.42	54.38
2016-05-23	30.93	54.38
2016-05-21	28.50	54.38
2016-05-24	20.82	54.38
2016-05-18	20.79	54.38
2016-05-25	20.32	54.38
2016-05-16	18.50	54.38
2016-05-27	10.51	54.38
2016-05-20	9.77	54.38

6 X_VALUE函数练习

练习54

- 需求，统计5月14日的不同网站收入情况，返回如下字段：
  - 网站id `website_id`，当日收入 `revenue`
  - 所有网站当日最高收入 `highest_revenue`
  - 所有网站当日最少收入 `lowest_revenue`

```
SELECT
  website_id,
  revenue,
  FIRST_VALUE(revenue) OVER(ORDER BY revenue) AS
lowest_revenue,
  LAST_VALUE(revenue) OVER(
    ORDER BY revenue
    ROWS BETWEEN UNBOUNDED PRECEDING
      AND UNBOUNDED FOLLOWING) AS highest_revenue
FROM statistics
WHERE day = '2016-05-14';
```

查询结果

website_id	revenue	lowest_revenue	highest_revenue
3	0.60	0.60	85.50
2	38.71	0.60	85.50
1	85.50	0.60	85.50

练习55

- 需求：统计id为1的网站的点击量，返回如下字段
  - 日期 `day` ,点击量 `clicks` ，5月点击量的中位数
  - 提示：5月一共31天，将点击量按顺序排列，第16位点击量即为中位数

```
SELECT
  day,
  clicks,
  NTH_VALUE(clicks, 16) OVER(
    ORDER BY clicks DESC
    ROWS BETWEEN UNBOUNDED PRECEDING
      AND UNBOUNDED FOLLOWING) AS `median`
FROM statistics
WHERE website_id = 1;
```

查询结果

day	clicks	median
2016-05-03	1545	526
2016-05-30	1382	526
2016-05-10	1029	526
2016-05-22	992	526
2016-05-19	857	526
2016-05-06	804	526
2016-05-31	796	526
2016-05-02	793	526
2016-05-08	782	526
2016-05-23	773	526
2016-05-28	645	526
2016-05-16	609	526
2016-05-18	594	526
2016-05-27	564	526
2016-05-24	553	526
2016-05-13	526	526
2016-05-29	446	526
2016-05-11	423	526
2016-05-25	416	526
……	……	……

练习56

- 需求：统计id为3的网站每天点击的情况，返回如下字段
    - 日期 `day` ，点击量 `clicks` ，最高点击量和当天点击量的比例 `ratio` （用整数表示）

```
SELECT
  day,
  clicks,
  ROUND(clicks / LAST_VALUE(clicks) OVER(
    ORDER BY clicks
    ROWS BETWEEN UNBOUNDED PRECEDING
      AND UNBOUNDED FOLLOWING) * 100) as `ratio`
FROM statistics
WHERE website_id = 3;
```

查询结果

day	clicks	round
2016-05-01	1	1
2016-05-02	1	1
2016-05-03	1	1
2016-05-04	1	1
2016-05-09	2	1
2016-05-05	2	1
2016-05-07	2	1
2016-05-06	3	2
2016-05-14	3	2
2016-05-12	4	2
2016-05-17	4	2
2016-05-13	4	2
2016-05-10	6	4
2016-05-21	7	4
2016-05-11	8	5
2016-05-08	8	5
2016-05-18	8	5
2016-05-15	9	5
2016-05-20	9	5
2016-05-16	11	7

小结

- `LEAD(x)` 和 `LAG(x)` 分别返回传入的列x对于当前行的下一行/前一行的值
- `LEAD(x,y)` 和 `LAG(x,y)` 分别返回传入的列x对于当前行的后y行/前y行的值

- `FIRST_VALUE(x)` 和 `LAST_VALUE(x)` 分别返回列x 的第一个值/最后一个值
- `NTH_VALUE(x,n)` 返回 x列的 第n个值
- `LAST_VALUE` 和 `NTH_VALUE` 通常要求把window frame修改成 `ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING`

练习57

- 业务背景：网站老板决定尝试新的商业模式：网站上只有一个广告位，对该广告位拍卖。 支付最高价的人将在一天之内在网站上展示他们的广告
- 该表非常简单：日期（天）支付的价格（价格）

price	day
33.03	2016-06-01
43.84	2016-06-02
37.25	2016-06-03
50.16	2016-06-04
26.63	2016-06-05
47.36	2016-06-06
32.02	2016-06-07
28.16	2016-06-08
38.12	2016-06-09
48.01	2016-06-10
27.56	2016-06-11
34.67	2016-06-12
37.09	2016-06-13
31.68	2016-06-14
22.61	2016-06-15
30.03	2016-06-16
22.13	2016-06-17
42.17	2016-06-18
21.81	2016-06-19
28.69	2016-06-20
34.72	2016-06-21
49.44	2016-06-22
25.82	2016-06-23
45.56	2016-06-24

price	day
48.21	2016-06-25
21.54	2016-06-26
37.76	2016-06-27
32.50	2016-06-28
26.66	2016-06-29
49.70	2016-06-30

- 需求：统计每日的拍卖价格和后一天的拍卖价格

```sql
SELECT
  day,
  price,
  LEAD(price) OVER(ORDER BY day) as `lead`
FROM advertisement;
```

查询结果

day	price	lead
2016-06-01	33.03	43.84
2016-06-02	43.84	37.25
2016-06-03	37.25	50.16
2016-06-04	50.16	26.63
2016-06-05	26.63	47.36
2016-06-06	47.36	32.02
2016-06-07	32.02	28.16
2016-06-08	28.16	38.12
2016-06-09	38.12	48.01
2016-06-10	48.01	27.56
2016-06-11	27.56	34.67
2016-06-12	34.67	37.09
2016-06-13	37.09	31.68
2016-06-14	31.68	22.61
2016-06-15	22.61	30.03
2016-06-16	30.03	22.13
2016-06-17	22.13	42.17
2016-06-18	42.17	21.81
2016-06-19	21.81	28.69
……	……	……

练习58

- 需求：统计每天的拍卖价格，7天前的拍卖价格，当天价格和7天前价格的差

```sql
SELECT
  day,
  price,
  LAG(price, 7) OVER(ORDER BY day) as lag7,
  price - LAG(price,7) OVER(ORDER BY day) as diff
FROM advertisement;
```

查询结果

day	price	lag	?column?
2016-06-01	33.03	null	null
2016-06-02	43.84	null	null
2016-06-03	37.25	null	null
2016-06-04	50.16	null	null
2016-06-05	26.63	null	null
2016-06-06	47.36	null	null
2016-06-07	32.02	null	null
2016-06-08	28.16	33.03	-4.87
2016-06-09	38.12	43.84	-5.72
2016-06-10	48.01	37.25	10.76
2016-06-11	27.56	50.16	-22.60
2016-06-12	34.67	26.63	8.04
2016-06-13	37.09	47.36	-10.27
2016-06-14	31.68	32.02	-0.34
2016-06-15	22.61	28.16	-5.55
2016-06-16	30.03	38.12	-8.09
2016-06-17	22.13	48.01	-25.88
2016-06-18	42.17	27.56	14.61
2016-06-19	21.81	34.67	-12.86
……	……	……	……

练习59

- 需求：查询每天的拍卖价格，所有价格中最高的，所有价格中最低的

```
SELECT
  day,
  price,
  FIRST_VALUE(price) OVER(ORDER BY price) AS lowest_price,
  LAST_VALUE(price) OVER(
    ORDER BY price
    ROWS BETWEEN UNBOUNDED PRECEDING
      AND UNBOUNDED FOLLOWING) AS highest_price
FROM advertisement;
```

查询结果

day	price	lowest_price	highest_price
2016-06-26	21.54	21.54	50.16
2016-06-19	21.81	21.54	50.16
2016-06-17	22.13	21.54	50.16
2016-06-15	22.61	21.54	50.16
2016-06-23	25.82	21.54	50.16
2016-06-05	26.63	21.54	50.16
2016-06-29	26.66	21.54	50.16
2016-06-11	27.56	21.54	50.16
2016-06-08	28.16	21.54	50.16
2016-06-20	28.69	21.54	50.16
2016-06-16	30.03	21.54	50.16
2016-06-14	31.68	21.54	50.16
2016-06-07	32.02	21.54	50.16
2016-06-28	32.50	21.54	50.16
2016-06-01	33.03	21.54	50.16
2016-06-12	34.67	21.54	50.16
2016-06-21	34.72	21.54	50.16
2016-06-13	37.09	21.54	50.16
2016-06-03	37.25	21.54	50.16
……	……	……	……