Final Project: Milestone 4

CS 327E
April 16, 2018
Announcements:

Today: Last regular class.
Today: Last quiz.
Next Friday: Demo Day in WAG 420. Schedule [link](#).
Discuss: Milestones 5 and 6 guidelines.
1) What makes traditional MapReduce suitable for batch processing?

A) The inputs to the Mapper are bounded / finite.

B) The inputs to the Reducer are bounded / finite.

C) The job is run at fixed time slices (e.g. now, hourly, daily, etc.)

D) All of the above.
2) What is the one crucial difference between a batch job and a streaming job?

A) The batch job processes larger collections of data.

B) The batch job goes through a multi-stage pipeline.

C) The event stream never ends.

D) None of the above.
3) Consider the Star Wars movies and their release timeline. The episode number is equivalent to _________ whereas the release year is equivalent to ___________.

Episode IV: 1977
Episode V: 1980
Episode VI: 1983
Episode I: 1999
Episode II: 2002
Episode III: 2005

A) Processing time; Event time
B) Event time; Processing time
4) The paper discusses 3 types of windows: Fixed, Sliding, and Sessions. Which notion of time are these windows based on?

A) Event time

B) Processing time

C) Neither
5) The paper discusses 3 options for handling straggler events that arrive after the window has been declared complete: **Discarding**, **Accumulating**, and **Accumulating & Retracting**. Which option(s) require the consumer to handle updated results for the windows?

A) Discarding  
B) Accumulating  
C) Accumulating & Retracting  
D) All of the above  
E) Only B and C
Case Expressions in SQL

- Conditional logic
- Since SQL:92 Standard
- Appear in SELECT clause
- Return scalar value for each record
- Return values of same type
- Used in SELECT statements
- Also used in UPDATE, INSERT, DELETE statements

General Form:

```
CASE
  WHEN c1 THEN e1
  WHEN c2 THEN e2
  ...
  WHEN cn THEN en
  [ELSE ed]
END
```
Case Expression Example

```sql
select listing_id,
    case
        when amenity_name = 'translation missing: en.hosting_amenity_49' then 'Unknown'
        when amenity_name = 'translation missing: en.hosting_amenity_50' then 'Unknown'
        when amenity_name is null then 'Unknown'
        when amenity_name = '' then 'Unknown'
        else amenity_name
    end as amenity_name
from `utcs-spr2018.austin.Amenity`
order by listing_id;
```

Results:

<table>
<thead>
<tr>
<th>Row</th>
<th>listing_id</th>
<th>amenity_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14913</td>
<td>TV</td>
</tr>
<tr>
<td>2</td>
<td>14913</td>
<td>Indoor fireplace</td>
</tr>
<tr>
<td>3</td>
<td>14913</td>
<td>Unknown</td>
</tr>
<tr>
<td>4</td>
<td>14913</td>
<td>Kitchen</td>
</tr>
<tr>
<td>5</td>
<td>14913</td>
<td>Heating</td>
</tr>
</tbody>
</table>
Another Case Expression Example

```sql
SELECT id, name, host_id, host_name, number_of_reviews,
       CASE
         WHEN number_of_reviews > 1000 THEN 'Many'
         WHEN number_of_reviews > 500  THEN 'Moderate'
         WHEN number_of_reviews >= 1   THEN 'Few'
         WHEN number_of_reviews = 0    THEN 'None'
       END AS reviews_label
FROM utcs-spr2018.austin.Summary_Listing
WHERE host_name IS NOT NULL;
```
Window Clause in SQL

- Informally called the **OVER** clause
- Since SQL:2003 Standard
- Rows split into partitions with **PARTITION BY** predicate
- Rows are sorted within each partition with **ORDER BY** predicate
- Window function applied to each row within partition
- Example functions: **ROW_NUMBER ()**, **RANK ()**

**General Form:**

```sql
SELECT c1, f() 
OVER ( 
    [PARTITION BY c3 
    ORDER BY c4] 
) 
FROM T1
```
Window Example: ROW_NUMBER

```sql
SELECT ROW_NUMBER() OVER() AS row_num, neighborhood_name, zipcode
FROM `utcs-spr2018.austin.Neighborhood`
WHERE zipcode IS NOT NULL;
```
Window Example: ROW_NUMBER

```sql
SELECT
    ROW_NUMBER() OVER() AS row_num,
    neighborhood_name,
    zipcode
FROM `utcs-spr2018.austin.Neighborhood`
WHERE zipcode IS NOT NULL;
```

```sql
SELECT
    ROW_NUMBER() OVER(ORDER BY neighborhood_name) AS row_num,
    neighborhood_name,
    zipcode
FROM `utcs-spr2018.austin.Neighborhood`
ORDER BY neighborhood_name;
```
Window Example: ROW_NUMBER

```
SELECT
    ROW_NUMBER() OVER(PARTITION BY neighborhood_name) AS row_num,
    neighborhood_name,
    zipcode
FROM
    `utcs-spr2018.austin.Neighborhood`;
```
Window Example: ROW_NUMBER

```sql
SELECT
  ROW_NUMBER() OVER(PARTITION BY neighborhood_name ORDER BY zipcode) AS row_num,
  neighborhood_name,
  zipcode
FROM
  'utcs-spr2018.austin.Neighborhood`
```
Window Example: RANK

```
SELECT id, host_id, price, 
    RANK() OVER(PARTITION BY host_id ORDER BY price) AS ranked_listing
FROM `utcs-spr2018.austin.Listing`
ORDER BY host_id, price;
```

<table>
<thead>
<tr>
<th>Row</th>
<th>id</th>
<th>host_id</th>
<th>price</th>
<th>ranked_listing</th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
<td>1737150</td>
<td>16920</td>
<td>75.0</td>
<td>1</td>
</tr>
<tr>
<td>44</td>
<td>9079111</td>
<td>16920</td>
<td>100.0</td>
<td>2</td>
</tr>
<tr>
<td>45</td>
<td>5684947</td>
<td>16920</td>
<td>125.0</td>
<td>3</td>
</tr>
<tr>
<td>46</td>
<td>5444799</td>
<td>16920</td>
<td>150.0</td>
<td>4</td>
</tr>
<tr>
<td>47</td>
<td>10385008</td>
<td>16920</td>
<td>400.0</td>
<td>5</td>
</tr>
<tr>
<td>48</td>
<td>13386694</td>
<td>17333</td>
<td>60.0</td>
<td>1</td>
</tr>
</tbody>
</table>
### Window Example: RANK and SUM

```sql
SELECT id, host_id, price,
    RANK() OVER(PARTITION BY host_id ORDER BY price) AS ranked_listing,
    SUM(price) OVER(PARTITION BY host_id ORDER BY price) AS running_total
FROM `utcs-spr2018.austin.Listing`
ORDER BY host_id, price;
```

<table>
<thead>
<tr>
<th>Row</th>
<th>id</th>
<th>host_id</th>
<th>price</th>
<th>ranked_listing</th>
<th>running_total</th>
</tr>
</thead>
<tbody>
<tr>
<td>43</td>
<td>1737150</td>
<td>16920</td>
<td>75.0</td>
<td>1</td>
<td>75.0</td>
</tr>
<tr>
<td>44</td>
<td>9079111</td>
<td>16920</td>
<td>100.0</td>
<td>2</td>
<td>175.0</td>
</tr>
<tr>
<td>45</td>
<td>5684947</td>
<td>16920</td>
<td>125.0</td>
<td>3</td>
<td>300.0</td>
</tr>
<tr>
<td>46</td>
<td>5444799</td>
<td>16920</td>
<td>150.0</td>
<td>4</td>
<td>450.0</td>
</tr>
<tr>
<td>47</td>
<td>10385008</td>
<td>16920</td>
<td>400.0</td>
<td>5</td>
<td>850.0</td>
</tr>
<tr>
<td>48</td>
<td>13386694</td>
<td>17333</td>
<td>60.0</td>
<td>1</td>
<td>60.0</td>
</tr>
</tbody>
</table>
Final Project Milestone 4

Cross-Dataset Joins: