SQL Queries
Full-Relation Operations

SQL Queries

```
SELECT [DISTINCT] [SUM | COUNT | AVG] result_table
FROM input_tables
[WHERE table_predicates]
[GROUP BY grouping_attributes
    [HAVING agg_condition]]
[ORDER BY sorting_attributes]
[UNION [ALL]] [INTERSECT] [EXCEPT]
```

Duplicate Elimination DISTINCT

```
SELECT [DISTINCT] result_table
FROM input_tables
[WHERE table_predicates]
```

- Transform the result from a multi-set (bag) to a set
- It is an expensive operation!

DISTINCT

- SELECT county

  FROM Cities_Population
- SELECT DISTINCT county

  FROM Cities_Population

- select maker

  from product
- select distinct maker

  from product
- select maker, type

  from product
- select distinct maker, type

  from product

Aggregates Functions

SELECT **[SUM | COUNT | AVG | MIN | MAX](agg_attributes)**

FROM input_tables

[WHERE table_predicates]

- The output table has a single tuple (row) that contains the result of the aggregate function
- When a single aggregate is computed, the result is a single table cell (1 row and 1 column)
- PANDAS describe() function

Aggregate Queries Cities

- PANDAS describe()
- SELECT count(county)

  FROM Cities_Population
- SELECT count(DISTINCT county)

  FROM Cities_Population
- select count(*) as cnt,

      min(pop_2010) as min_pop,

      avg(pop_2010) as avg_pop,

      max(pop_2010) as max_pop

  from Cities_Population

- select max(pop_2010-pop_2000) as max_pop_increase,

  min(pop_2010-pop_2000) as max_pop_decrease,

  avg(pop_2010-pop_2000) as avg_pop_increase

  from Cities_Population

Aggregate Queries Computers

- select count(*)

  from product

  where maker = ‘A’
- select AVG(price)

  from PC
- select MIN(price), AVG(price), MAX(price)

  from laptop
- select min(speed), min(hd)

  from pc

  where price > 1000
- select count (distinct maker)

  from product

  where type = ‘pc’

GroupBy Aggregates

SELECT **grouping_atts, [SUM | COUNT | AVG | MIN | MAX](agg_attributes)**

FROM input_tables

[WHERE table_predicates]

[GROUP BY grouping_atts

[HAVING agg_condition]]

- Split input table into groups of tuples that have the same value for the grouping_atts
- Compute the aggregate functions for the tuples in every group
- Output a **single** tuple for every group: (grouping_atts, agg_functions)
- **HAVING** is a WHERE applied on the output
- WHERE is applied before the grouping

GroupBy Aggregates Cities

- select county,

  count(*) as no_city,

  min(pop_2010) as min_pop,

  avg(pop_2010) as avg_pop,

  max(pop_2010) as max_pop,

  sum(pop_2010) as total_pop

  from Cities_Population

  group by county

- select county,

  count(*) as no_city,

  min(pop_2010) as min_pop,

  avg(pop_2010) as avg_pop,

  max(pop_2010) as max_pop,

  sum(pop_2010) as total_pop

  from Cities_Population

  group by county

  having no_city >= 10

  order by no_city desc, total_pop desc

GroupBy Aggregates Computers

- select speed, avg(price) as avg_price

  from pc

  group by speed
- select speed, avg(price) as avg_price

  from pc

  where speed > 2

  group by speed

- select maker, count (distinct model)

  from product

  group by maker
- select maker, count (distinct model)

  from product

  where type = 'pc'

  group by maker
- select maker, count (distinct model) as models

  from product

  where type = 'pc'

  group by maker

  having models >= 3

Examples

- Cities
- Computers
- TPCH