Manual showcase (prefSQL)

* Basic query

**SELECT**

t1.id, t1.title

**FROM** cars t1

LEFT OUTER JOIN colors ON t1.color\_id = colors.ID

LEFT OUTER JOIN bodies ON t1.body\_id = bodies.ID

**SKYLINE OF** <My Preferences>

**ORDER BY** <My Sort Clause>

* Preferences (Skyline Of)
  + Price
    - Lower price is better

**t1.price LOW**

* + - Higher price is better

**t1.price HIGH**

* + - Closer price to 10‘000 is better

**t1.price AROUND 10000**

* + Colour
    - Red is better than others. Others are indifferent (= equally preferred).

**colors.name ('rot' >> OTHERS EQUAL)**

* + - Red is better than others. Others are not comparable.

**colors.name ('rot' >> OTHERS INCOMPARABLE)**

* + - Pink is better than red and black. Red and black are indifferent. Red and black are better than beige. Beige is better than other colours. Other colours are not comparable.

**colors.name ('pink' >> {'rot', 'schwarz'} >> 'beige' >> OTHERS INCOMPARABLE)**

* + Chassis
    - Bus is better than compact car

**bodies.name ('Bus' >> 'Kleinwagen')**

* + - Bus is better than compact car. Compact car is better than other cars. Other cars are equally preferred.

**bodies.name ('Bus' >> 'Kleinwagen' >> OTHERS EQUAL)**

* + - Compact car is better than bus; Bus is better than station wagon.; Station wagon is better than scooter; Scooter is better than others; Others are indifferent; Others are better than pick-up.

**bodies.name ('Kleinwagen' >> 'Bus' >> 'Kombi' >> 'Roller' >> OTHERS EQUAL >> 'Pick-Up')**

* Sort (Order By)
  + With the sort clause the result can be ordered. In general every sort clause like in normal SQL is possible. PrefSQL only offers some shorter syntax
  + Use normal SQL syntax to sort for certain attributes

**ORDER BY t1.price, t1.mileage**

* + Sort categories (Use the PrefSQL syntax)

**ORDER BY colors.name ('pink' >> OTHERS EQUAL)**

* + Sort for best tuples. (Sum of the ranking of your attributes)
    - All attributes from the skyline (Every attribute in the SKYLINE OF clause)

**ORDER BY SUM\_RANK()**

* + - Ranking with own attributes (Not implemented yet)

**ORDER BY SUM\_RANK(t1.price ASC, t1.mileage ASC)**

* + Sort diverse (From each attribute the best ranking is used)
    - Alle attributes from the skyline (Every attribute in the SKYLINE OF clause)

**ORDER BY BEST\_RANK()**

* + - Own attributes (Not implemented yet)

**ORDER BY BEST\_RANK(t1.price ASC, t1.mileage ASC)**

* + Sort randomly (Use SQL Syntax)

**ORDER BY NEWID()**

* Example query

**SELECT**

t1.id, t1.title, t1.price, colors.name, bodies.name

**FROM** cars t1

LEFT OUTER JOIN colors ON t1.color\_id = colors.ID

LEFT OUTER JOIN bodies ON t1.body\_id = bodies.ID

**SKYLINE OF** t1.price LOW, colors.name (‘rot’ >> OTHERS EQUAL), bodies.name (‘Bus’ >> ‘Kleinwagen’ >> OTHERS EQUAL)

**ORDER BY** BEST\_RANK()

* eCommerce Tables
  + Cars\_small 460 records
  + Cars\_medium 920 records
  + Cars\_large 5520 records
  + Cars\_superlarge 55208 records