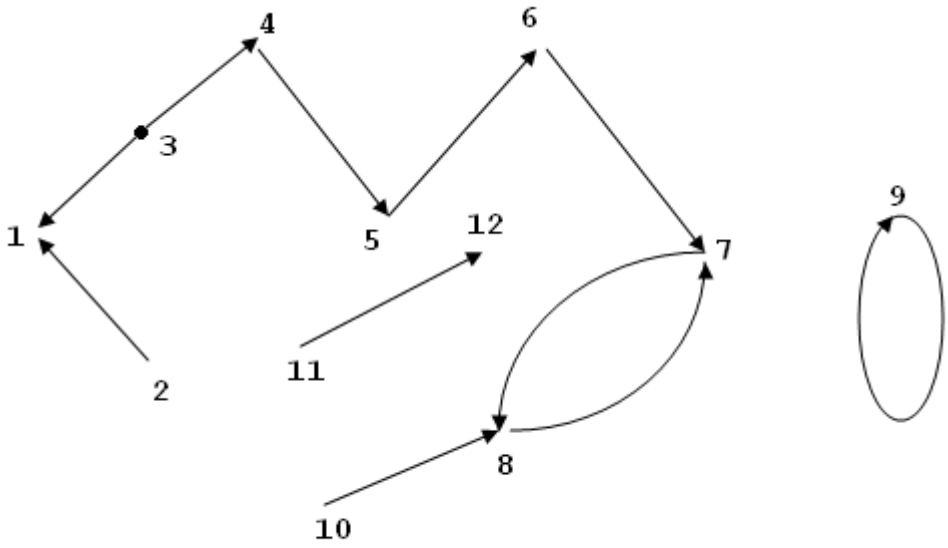
All queries should be implemented in pure SQL and without using any undocumented features.

***

Table contains edges for the directed graphs. Please write a query that returns all the nodes that are linked to the given node.

For example, for below data, result for the node 1 should be the following:




```
SQL> with graph as
  2  (select 1 id, 2 parent_id from dual
  3  union all select 1 id, 3 parent_id from dual
  4  union all select 4 id, 3 parent_id from dual
  5  union all select 5 id, 4 parent_id from dual
  6  union all select 6 id, 5 parent_id from dual
  7  union all select 7 id, 6 parent_id from dual
  8  union all select 8 id, 7 parent_id from dual
  9  union all select 7 id, 6 parent_id from dual
 10  union all select 8 id, 10 parent_id from dual
 11  union all select 9 id, 9 parent_id from dual
 12  union all select 12 id, 11 parent_id from dual)
...
 20  /

        ID
----------
         1
         2
         3
         4
         5
         6
         7
         8
        10
```

***

Table contains information about user sessions. Please write a query that returns the maximum number of simultaneous sessions for each user and the earliest time when session count was maximal. Even though sample data contains only two users, in general case the number of users is not fixed.

For the below data, result should be the following:

```
SQL> with log as
  2  (select 'U1' username, date '2013-08-08'+1/24 logon_time,
  3   date '2013-08-08'+10/24 logoff_time from dual
  4  union all select 'U1' username, date '2013-08-08'+6/24 logon_time,
  5   date '2013-08-08'+14/24 logoff_time from dual
  6  union all select 'U1' username, date '2013-08-08'+4/24 logon_time,
  7   date '2013-08-08'+12/24 logoff_time from dual
  8  union all select 'U1' username, date '2013-08-08'+8/24 logon_time,
  9   date '2013-08-08'+17/24 logoff_time from dual
 10  union all select 'U1' username, date '2013-08-08'+16/24 logon_time,
 11   date '2013-08-08'+18/24 logoff_time from dual
 12  union all select 'U1' username, date '2013-08-08'+9/24 logon_time,
 13   date '2013-08-08'+16/24 logoff_time from dual
 14  union all select 'U2' username, date '2013-08-08'+1/24 logon_time,
 15   date '2013-08-08'+3/24 logoff_time from dual
 16  union all select 'U2' username, date '2013-08-08'+2/24 logon_time,
 17   date '2013-08-08'+12/24 logoff_time from dual
 18  union all select 'U2' username, date '2013-08-08'+11/24 logon_time,
 19   date '2013-08-08'+13/24 logoff_time from dual
 20  union all select 'U2' username, date '2013-08-08'+10/24 logon_time,
 21   date '2013-08-08'+14/24 logoff_time from dual)
...
 37  /

US CNT_SESSIONS TIME
-- ------------ -------------------
U1            5 08.08.2013 09.00.00
U2            3 08.08.2013 11.00.00
```

***

There is a fact table risk (value number, created timestamp, expired timestamp) populated in batches. For the inserted records the process populates “created” with batch timestamp and “expired” with null. For the expired records the process marks “expired” with batch timestamp without affecting “created”. Every batch may create new records, expire existing records or do the both. The goal is to find total value for created and expired records for every batch.

For the below data

```
with risk(created, expired, value) as
(
select timestamp '2022-01-01 00:00:01', timestamp '2022-01-01 00:00:02', 1 from dual
union all
select timestamp '2022-01-01 00:00:01', timestamp '2022-01-01 00:00:03', 10 from dual
union all
select timestamp '2022-01-01 00:00:01', null, 100 from dual
union all
select timestamp '2022-01-01 00:00:02', null, 2000 from dual
union all
select timestamp '2022-01-01 00:00:02', timestamp '2022-01-01 00:00:03', 1000 from dual
union all
select timestamp '2022-01-01 00:00:02', timestamp '2022-01-01 00:00:04', 10000 from dual
union all
select timestamp '2022-01-01 00:00:04', null, 100000 from dual
)
...
/
```

The expected result is following

```
TSTAMP                              TOTAL_CREATED TOTAL_EXPIRED
----------------------------------- ------------- -------------
01/01/2022 00:00:01.000000000               111
01/01/2022 00:00:02.000000000             13000             1
01/01/2022 00:00:03.000000000                               1010
01/01/2022 00:00:04.000000000            100000         10000
```

***

For given master and detail tables please write a query that returns sum of values with grouping by grp as well as total summary of master values in the first row. Only one scan of master table is allowed.

```
SQL> with
  2  master as
  3  (select 1 as id_m, 111 as value  from dual union all
  4  select 2 as id_m, 222 as value  from dual union all
  5  select 3 as id_m, 333 as value  from dual union all
  6  select 4 as id_m, 444 as value  from dual union all
  7  select 5 as id_m, 555 as value  from dual union all
  8  select 6 as id_m, 666 as value  from dual),
  9  detail as
 10  (select 1 as id_m, 1 as grp from dual union all
 11  select 1 as id_m, 2 as grp from dual union all
 12  select 1 as id_m, 4 as grp from dual union all
 13  select 2 as id_m, 3 as grp from dual union all
 14  select 2 as id_m, 4 as grp from dual union all
 15  select 3 as id_m, 1 as grp from dual union all
 16  select 3 as id_m, 3 as grp from dual union all
 17  select 3 as id_m, 5 as grp from dual)
...
 24  /

       GRP     ALLSUM
---------- ----------
                 2331
         1         444
         2         111
         3         555
         4         333
         5         333
```

***

Table contains nodes for different trees. Write a query that returns signed nodes along with their parents on first three levels. "connect by" clause can be used only once. Parent ID for the root nodes is not necessarily blank, below is just a sample data.

```
SQL> with tree as
  2  (select 3 id, 1 parent_id, 0 sign from dual
  3  union all select 4 id, 2 parent_id, 0 sign from dual
  4  union all select 5 id, 2 parent_id, 0 sign from dual
  5  union all select 6 id, 3 parent_id, 0 sign from dual
  6  union all select 7 id, 3 parent_id, 0 sign from dual
  7  union all select 8 id, 3 parent_id, 0 sign from dual
  8  union all select 9 id, 4 parent_id, 0 sign from dual
  9  union all select 10 id, 4 parent_id, 1 sign from dual
 10  union all select 11 id, 7 parent_id, 1 sign from dual
 11  union all select 12 id, 7 parent_id, 0 sign from dual
 12  union all select 13 id, 9 parent_id, 0 sign from dual
 13  union all select 14 id, 9 parent_id, 1 sign from dual
 14  union all select 15 id, 9 parent_id, 1 sign from dual
 15  union all select 2 id, null parent_id, 0 sign from dual
 16  union all select 1 id, null parent_id, 0 sign from dual)
 ...
 30  /

SIGNED_NODE    ID_LVL1    ID_LVL2    ID_LVL3
----------- ---------- ---------- ----------
         10          2          4         10
         11          1          3          7
         14          2          4          9
         15          2          4          9
```