home_credit_test

March 27, 2022

1 Import libriaries and update sqlite3 version

[ ]:
```
!gdown --id 1BSHIKQ7rFw5BpTq5nw1UZfjPK_7Mpnbi
!mv _sqlite3.cpython-37m-x86_64-linux-gnu.so /usr/lib/python3.7/lib-dynload/
```

```
Downloading…
From: https://drive.google.com/uc?id=1BSHIKQ7rFw5BpTq5nw1UZfjPK_7Mpnbi
To: /content/_sqlite3.cpython-37m-x86_64-linux-gnu.so
100% 6.50M/6.50M [00:00<00:00, 96.8MB/s]
```

[ ]:
```
import pandas as pd
import numpy as np
import sqlite3
```

[ ]:
```
sqlite3.version
```

[ ]:
```
'2.6.0'
```

[ ]:
```
con = sqlite3.connect('db')
```

[ ]:
```
def select(sql):
  return pd.read_sql(sql,con)
```

2 Task 1 from excel list 'emp'. Create tables.

[ ]:
```
employee = pd.DataFrame(
    {
        'id_emp': [i for i in range(1,13)],
        'emp_name': ['Alex', 'Sasha', 'Gleb', 'Andrew', 'Kate', 'Jess', 'Hank',␣
↪'Karen', 'Rankl', 'Marsy', 'Stew', 'Bill']
    }
)
employee.to_sql('employee', con, index=False, if_exists='replace')
```

[ ]:
```
emp_prem = pd.DataFrame(
    {
        'id_emp': [1, 2, 1, 3, 4, 8, 2, 10, 9, 6, 11, 2],
        'month': ['2022-01-01', '2022-02-01', '2022-03-01', '2021-12-01',␣
↪'2022-01-01', '2022-03-01', '2022-01-01', '2021-12-01', '2022-02-01',␣
↪'2021-12-01', '2021-12-01', '2022-03-01'],
        'premium': [3000, 5000, 1000, 3500, 4000, 2000, 7000, 6500, 3800, 2000,␣
↪6500, 1000]
    }
)
emp_prem['month'] = pd.to_datetime(emp_prem['month'],format='%Y-%m-%d')
emp_prem.to_sql('emp_prem', con, index=False, if_exists='replace')
```

2.1 Display all employees with the highest bonus in each month

[ ]:
```
sql = '''
select
  *
from (
  select
    t.*
    , p.month
    , p.premium
    , rank() over (partition by p.month order by premium desc) as rnk
  from employee as t
  left join emp_prem as p on t.id_emp = p.id_emp
) as t
where t.rnk = 1 and t.premium is not null
'''
select(sql)
```

[ ]:

	id_emp	emp_name	month	premium	rnk
0	10	Marsy	2021-12-01 00:00:00	6500	1
1	11	Stew	2021-12-01 00:00:00	6500	1
2	2	Sasha	2022-01-01 00:00:00	7000	1
3	2	Sasha	2022-02-01 00:00:00	5000	1
4	8	Karen	2022-03-01 00:00:00	2000	1

2.2 For those employees who have the maximum bonus in a given month, apply a 30% bonus, if several employees have the same maximum bonus, then do not apply the bonus

[ ]:
```
sql = '''
with max_premium as (
  select
    *
  from (
    select
      t.*
      , p.month
      , p.premium
      , rank() over (partition by p.month order by premium desc) as rnk
    from employee as t
    left join emp_prem as p on t.id_emp = p.id_emp
  ) as t
  where t.rnk = 1 and t.premium is not null
),

count_max_prem as (
  select
    t.month
    , sum(rnk) as cnt_max
  from max_premium as t
  group by t.month
)
select
  t.*
  , cmp.cnt_max
  , case
      when cmp.cnt_max > 1 then t.premium
    else t.premium*1.3 end as plus_premium
from max_premium as t
left join count_max_prem as cmp on t.month = cmp.month

'''
select(sql)
```

[ ]:

	id_emp	emp_name	month	premium	rnk	cnt_max	plus_premium
0	10	Marsy	2021-12-01 00:00:00	6500	1	2	6500.0
1	11	Stew	2021-12-01 00:00:00	6500	1	2	6500.0
2	2	Sasha	2022-01-01 00:00:00	7000	1	1	9100.0
3	2	Sasha	2022-02-01 00:00:00	5000	1	1	6500.0
4	8	Karen	2022-03-01 00:00:00	2000	1	1	2600.0

3 Task 2 from excel list 'credit'. Create tables.

[ ]:
```
credit = pd.DataFrame(
    {
        'id_credit': [i for i in range(1,21)],
        'id_emp': [2, 2, 1, 3, 4, 8, 9, 3, 7, 6, 11, 12, 10, 1, 3, 8, 9, 11,␣
↪12, 7],
        'id_seller_place': [25, 36, 44, 52, 34, 63, 39, 39, 25, 44, 52, 63, 25,␣
↪63, 34, 36, 34, 63, 52, 25],
        'date_sale': ['2021-12-03', '2021-12-14', '2021-12-18', '2021-12-25',␣
↪'2021-12-03', '2021-12-18', '2022-01-18', '2022-01-25',␣
↪'2022-01-27','2022-02-10',
                      '2022-02-18', '2022-02-24', '2022-03-07', '2022-03-19',␣
↪'2022-03-05', '2022-02-28', '2022-03-21', '2022-03-27', '2022-01-29',␣
↪'2022-01-07'],
        'volume': [10000, 12000, 15000, 20000, 150000, 70000, 80000, 200000,␣
↪40000, 90000,
                   30000, 200000, 100000, 130000, 70000, 55000, 90000, 130000,␣
↪50000, 15000],
        'payment_num': [6, 10, 12, 8, 8, 9, 10, 11, 15, 12,
                        7, 6, 11, 9, 9, 11, 12, 8, 9, 6],
        'Status': ['failure', 'approved', 'approved', 'approved', 'approved',␣
↪'failure', 'failure', 'failure', 'approved', 'approved',
                   'approved', 'failure', 'failure', 'approved', 'approved',␣
↪'approved', 'failure', 'failure', 'approved', 'approved']
    }
)
credit['date_sale'] = pd.to_datetime(credit['date_sale'],format='%Y-%m-%d')
credit.to_sql('credit', con, index=False, if_exists='replace')
```

[ ]:
```
seller_place = pd.DataFrame(
    {
        'id_seller_place': [25, 36, 44, 52, 34, 63, 39],
        'city': ['Moscow', 'St.Petersburg', 'Kazan', 'Kamchatka', 'Sahalin',␣
↪'California', 'Tula']
    }
)
seller_place.to_sql('seller_place', con, index=False, if_exists='replace')
```

[ ]:
```
employee = pd.DataFrame(
    {
        'id_emp': [i for i in range(1,13)],
        'emp_name': ['Alex', 'Sasha', 'Gleb', 'Andrew', 'Kate', 'Jess', 'Hank',␣
↪'Karen', 'Rankl', 'Marsy', 'Stew', 'Bill'],
        'status': ['a', 'a', 'n', 'a', 'n', 'a', 'a', 'a', 'n', 'a', 'a', 'n']
    }
```

```
)
employee.to_sql('employee', con, index=False, if_exists='replace')
```

3.1 Download the top 10 (2) employees with the highest sales (by total) in February 2016 (2022)

[ ]:
```
sql = '''
select
  t.id_emp
  , t.emp_name
  , sum(cr.volume) as sum_vol
from employee as t
left join credit as cr on t.id_emp = cr.id_emp
where date(cr.date_sale, 'start of month') = '2022-02-01' and cr.status =␣
 ↪'approved'
group by
  t.id_emp
  , t.emp_name
order by sum_vol desc
limit 2
'''
select(sql)
```

[ ]:

	id_emp	emp_name	sum_vol
0	6	Jess	90000
1	8	Karen	55000

3.2 Download the average loan size, and the weighted average (by volume) average term in the city of Voronezh (California) in 2016 (2022)

3.2.1 I doubt that I correctly understood the calculation of the metric, so I will describe the logic:

1. First, I calculated the average size of loans issued for all regions for 2022;
2. Next, I found the total volume of loans issued for 2022;
3. In the final table, I displayed the product of the average term and the share of the average loan in the region in the total volume of loans issued.

[ ]:
```
sql = '''
with mean_credit_size as (
  select
    sp.id_seller_place
    , avg(t.volume) as avg_vol
  from credit as t
  left join seller_place as sp on sp.id_seller_place = t.id_seller_place
```

```
  where (cast(strftime('%Y', t.date_sale) as int)) = 2022
    and t.status = 'approved'
  group by
    sp.id_seller_place
),

sum_credit_size as (
  select
    sum(volume) as sum_vol
  from credit as t
  where (cast(strftime('%Y', t.date_sale) as int)) = 2022
    and t.status = 'approved'
),

weighted_mean as (
  select
    t.id_seller_place
    , sp.city
    , scs.sum_vol
    , cs.avg_vol
    , avg(t.payment_num) as avg_term
    , avg(t.payment_num) * (sum(cast(cs.avg_vol as float)) / sum(cast(scs.
 ↪sum_vol as float))) as weighted_term
  from credit as t
  left join seller_place as sp on sp.id_seller_place = t.id_seller_place
  left join mean_credit_size as cs on cs.id_seller_place = t.id_seller_place
  left join sum_credit_size as scs on 1=1
  where (cast(strftime('%Y', t.date_sale) as int)) = 2022
    and sp.id_seller_place = 63
  group by
    t.id_seller_place
    , sp.city
)
select * from weighted_mean
'''
select(sql)
```

[ ]:

	id_seller_place	city	sum_vol	avg_vol	avg_term	weighted_term
0	63	California	480000	130000.0	7.666667	2.076389

3.3 Calculate the percentage of loan approval in January 2016 (2022) by city

[ ]:
```
sql = '''
select
  t.city
  , avg(t.num_status) as approved_pct
```

```
from (
  select
    sp.city
    , t.Status
    , case
        when t.Status = 'approved' then 1
        else 0
      end as num_status
  from credit as t
  left join seller_place as sp on sp.id_seller_place = t.id_seller_place
  where date(t.date_sale, 'start of month') = '2022-01-01'
) as t
group by
  t.city
'''
select(sql)
```

[ ]:

	city	approved_pct
0	Kamchatka	1.0
1	Moscow	1.0
2	Tula	0.0

3.4 For employees who had no sales in 2016 (2022), change status to inactive 'n'

[ ]:
```
sql = '''
with approved_sales as (
select
  t.id_emp
  , t.status
from employee as t
left join credit as cr on t.id_emp = cr.id_emp
where (cast(strftime('%Y', cr.date_sale) as int)) = 2022
  and cr.status = 'approved'
)

select
  t.id_emp
  , aps.status
  , case
      when aps.status is null then 'n'
      else aps.status
  end as correct_status
from employee as t
left join approved_sales as aps on t.id_emp = aps.id_emp
'''
```

```
select(sql)
```

[ ]:
```
    id_emp status correct_status
0        1      a              a
1        2   None              n
2        3      n              n
3        4   None              n
4        5   None              n
5        6      a              a
6        7      a              a
7        7      a              a
8        8      a              a
9        9   None              n
10      10   None              n
11      11      a              a
12      12      n              n
```

4 Task 3 from excel list 'Risk'. Create tables.

[ ]:
```
transaction = pd.DataFrame(
    {
        'id_card': [4, 4, 4, 4, 4, 4, 1, 5, 5, 2,
                    2, 2, 3, 3, 3, 1, 1, 3, 5, 1,
                    4, 4, 5, 1, 1, 5, 5, 5, 5, 5],
        'date_transac': ['2021-12-03', '2021-12-14', '2021-12-18',␣
 ↪'2021-12-25', '2021-12-03', '2021-12-18', '2022-01-18', '2022-01-25',␣
 ↪'2022-01-27','2022-02-10',
                         '2022-02-18', '2022-02-24', '2022-03-07', '2022-03-19',␣
 ↪'2022-03-05', '2022-02-28', '2022-03-21', '2022-03-27', '2022-01-29',␣
 ↪'2022-01-07',
                          '2021-12-25', '2021-12-03', '2022-02-18',␣
 ↪'2022-01-18', '2022-01-25', '2022-02-21', '2022-02-01', '2022-03-11',␣
 ↪'2022-03-08', '2022-03-24'],
        'cash': [20000, 5000, 1500, 700, 500, 1200, 1400, 2100, 1800, 3500,
                 200, 300, 1200, 1000, 700, 100, 50, 550, 320, 1210,
                 300, 230, 560, 790, 800, 15000, 18000, -1000, -500, 100]
    }
)
transaction['date_transac'] = pd.
 ↪to_datetime(transaction['date_transac'],format='%Y-%m-%d')
transaction.to_sql('transaction_t', con, index=False, if_exists='replace')
```

[ ]:
```
card = pd.DataFrame(
    {
        'id_card': [i for i in range(1,6)],
```

```
        'date_activ': ['2022-01-01', '2022-02-01', '2022-03-01', '2021-11-15',␣
↪'2022-01-10'],
        'limit_t': [30000, 50000, 100000, 35000, 40000]
    }
)
card['date_activ'] = pd.to_datetime(card['date_activ'],format='%Y-%m-%d')
card.to_sql('card', con, index=False, if_exists='replace')
```

4.1 Withdraw cards that had at least one transaction within 30 (15) days from the moment the card was activated

[ ]:
```
sql = '''
with first_transaction as (
  select * from (
    select
      id_card
      , date_transac
      , row_number() over (partition by id_card order by date_transac) as␣
↪first_trans
    from transaction_t as t
  ) as t
  where first_trans = 1
)

select
  t.*
  , ft.date_transac as first_date_transac
  , cast((julianday(ft.date_transac) - julianday(t.date_activ)) as integer) as␣
↪diff
from card as t
left join first_transaction as ft on t.id_card = ft.id_card
where diff <= 15
'''
select(sql)
```

[ ]:

	id_card	date_activ	limit_t	first_date_transac	diff
0	1	2022-01-01 00:00:00	30000	2022-01-07 00:00:00	6
1	2	2022-02-01 00:00:00	50000	2022-02-10 00:00:00	9
2	3	2022-03-01 00:00:00	100000	2022-03-05 00:00:00	4
3	5	2022-01-10 00:00:00	40000	2022-01-25 00:00:00	15

4.2 Withdraw cards whose transaction amount is more than 80% of the card limit

[ ]:
```
sql = '''
with transaction_sum as (
  select
    t.id_card
    , t.limit_t
    , sum(tr.cash) as transaction_sum
  from card as t
  left join transaction_t as tr on t.id_card = tr.id_card
  group by
    t.id_card
    , t.limit_t
)

select * from (
  select
    id_card
    , (sum(cast(transaction_sum as float)) / sum(cast(limit_t as float))) as␣
 ↪ratio
  from transaction_sum
  group by id_card
) as t
where t.ratio > 0.8
'''
select(sql)
```

[ ]:

	id_card	ratio
0	4	0.840857
1	5	0.909500

4.3 Increase the card limit by 50% for customers who have had more than 10 (2) transactions per month during the last 3 (2) months

[ ]:
```
sql = '''
with count_transactions as (
  select
    t.*
    , cast((julianday('now') - julianday(t.month)) as integer) as diff
  from (
    select
      id_card
      , date(t.date_transac, 'start of month', '+1 month', '-1 day') as month
      , count(id_card) as cnt_trans
      , case
```

```
          when count(id_card) > 2 then 1
          else 0
      end as true_false
    from transaction_t as t
    group by
      id_card
      , month
  ) as t
),

necessary_card_id as (
  select
    t.id_card
  from count_transactions as t
    where t.diff < 32
    group by t.id_card
    having sum(true_false) >= 2
)

select
  t.*
  , case
      when nci.id_card is not null then limit_t*1.5
      else limit_t
  end as new_limit
from card as t
left join necessary_card_id as nci on t.id_card = nci.id_card
'''
select(sql)
```

[ ]:

	id_card	date_activ	limit_t	new_limit
0	1	2022-01-01 00:00:00	30000	30000.0
1	2	2022-02-01 00:00:00	50000	50000.0
2	3	2022-03-01 00:00:00	100000	100000.0
3	4	2021-11-15 00:00:00	35000	35000.0
4	5	2022-01-10 00:00:00	40000	60000.0

4.4 Reset the limit on cards for clients who have not had (less than 3) transactions during the last 12 (2) months

[ ]:
```
sql = '''
with count_transactions as (
  select
    t.*
    , cast((julianday('now') - julianday(t.month)) as integer) as diff
  from (
```

```
    select
      id_card
      , date(t.date_transac, 'start of month', '+1 month', '-1 day') as month
      , count(id_card) as cnt_trans
      , case
          when count(id_card) > 2 then 1
          else 0
      end as true_false
    from transaction_t as t
    group by
      id_card
      , month
  ) as t
),

necessary_card_id as (
  select
    t.id_card
  from count_transactions as t
    where t.diff < 32
    group by t.id_card
    having sum(cnt_trans) < 3
)

select
  t.*
  , case
      when nci.id_card is not null then limit_t*0.0
      else limit_t
  end as new_limit
from card as t
left join necessary_card_id as nci on t.id_card = nci.id_card
'''
select(sql)
```

[ ]:

	id_card	date_activ	limit_t	new_limit
0	1	2022-01-01 00:00:00	30000	0.0
1	2	2022-02-01 00:00:00	50000	50000.0
2	3	2022-03-01 00:00:00	100000	100000.0
3	4	2021-11-15 00:00:00	35000	35000.0
4	5	2022-01-10 00:00:00	40000	40000.0