Задача 1.

Какая часть дохода по каждому контракту приходится на первый день месяца*?

CONRACT_REVENUE

CONTRACT_ID	EVENT_DT	REVENUE
101	01.08.2020	200
101	01.09.2020	200
102	01.09.2020	300
103	01.09.2020	200
103	10.09.2020	150

```
CREATE TABLE CONRACT_REVENUE(
   CONTRACT_ID int,
   EVENT_DT datetime,
   REVENUE int
);

INSERT INTO CONRACT_REVENUE
values
   (101,  '01.08.2020',       200),
          (101,     '01.09.2020',       200),
          (102,     '01.09.2020',       300),
          (103,     '01.09.2020',       200),
          (103,     '10.09.2020',       150);

select * from CONRACT_REVENUE
where EVENT_DT like '01.%';
```

Задача 2.

Вывести количество контрактов по каждому продукту на данный момент, если START_DT это дата начала действия контракта, END_DT – дата окончания.

CONTRACTS_PRODUCTS

CONTRACT_ID	PRODUCT_ID	START_DT	END_DT
101	1	01.01.2020	01.07.2020
101	3	01.07.2020	01.09.2020
101	4	01.09.2020	31.12.2099
102	2	01.08.2020	31.12.2099
103	4	01.01.2020	31.12.2099

PRODUCT

PRODUCT_ID	PRODUCT_N
1	o2 Blue
2	o2 Blue M
3	o2 Free
4	o2 Free M
5	o2 Free

*По каждой задаче достаточно написать текст запроса, без реализации

```
CREATE TABLE
CONTRACTS_PRODUCTS (
CONTRACT_ID int,
PRODUCT_ID int,
START_DT  datetime,
END_DT datetime
);

INSERT INTO
CONTRACTS_PRODUCTS
values
   (101, 1, '01.01.2020', '01.07.2020'),
          (101, 3, '01.07.2020', '01.09.2020'),
          (101, 4, '01.09.2020', '31.12.2099'),
          (102, 2, '01.08.2020',  '31.12.2099'),
          (103, 4, '01.01.2020', '31.12.2099');

CREATE TABLE PRODUCT (
PRODUCT_ID int,
PRODUCT_NAME  varchar(255)
);

INSERT INTO PRODUCT
values
   (1, 'o2 Blue S'),
          (2, 'o2 Blue M'),
          (3, 'o2 Free S'),
          (4, 'o2 Free M'),
          (5, 'o2 Free L');

SELECT PRODUCT_ID,
COUNT(DISTINCT CONTRACT_ID) as
CONTRACTS
from CONTRACTS_PRODUCTS
where CONTRACT_ID in (
--активные контракты
SELECT DISTINCT(CONTRACT_ID)
from CONTRACTS_PRODUCTS
where END_DT > DATE('now') and
START_DT < DATE('now'))
group by PRODUCT_ID;
```

Задача 3.

Имеется таблица Sales, структура и пример содержания:

Manager	Product	Sale	Date
1	10	500	10.01.2020
1	20	300	20.01.2020
1	30	500	30.12.2019
2	10	400	10.02.2020
2	20	500	11.02.2020
2	30	500	11.02.2020
3	20	500	30.12.2019
3	30	300	10.01.2020

Необходимо написать запрос, выбирающий из этой таблицы для каждого менеджера (manager) одну строку с максимальной суммой продажи (sale), если строк с максимальной суммой продажи для менеджера несколько, то нужно выбрать ту из них, в которой дата продажи (date) больше, если для строк с максимальными суммами продажи и даты продажи одинаковые, то любую из них.

Результат запроса должен содержать четыре колонки как в исходной таблице:

Manager	Product	Sale	Date

```
create TABLE  Sales
(
Manager int,
Product int,
Sale int,
Date datetime
);

INSERT INTO  Sales VALUES
(1,10,500,'10.01.2020'),
(1, 20, 300, '20.01.2020'),
(1, 30, 500, '30.12.2019'),
(2, 10, 400, '10.02.2020'),
(2, 20, 500, '11.02.2020'),
(2, 30, 500, '11.02.2020'),
(3, 20, 500, '30.12.2019'),
(3, 30, 300, '10.01.2020');
```

```
--drop TABLE Rank_table_blank;
-- создаем колонку для ранжирования Sales
create temp table Rank_table_blank as
select
null as rank,
Manager,	Product,	Sale,	Date
from Sales
order by Sale DESC ;
--drop TABLE Rank_table;
-- ранжируем Sales по убыванию и заполняем колонку sale_rank
create temp table Rank_table as
select
dense_rank() over (partition by manager order by sale desc) as sale_rank,
Manager,	Product,	Sale,	Date
from Rank_table_blank
window w as (order by  Sale DESC)
order by sale_rank;

-- создаем колонку для ранжирования Date
--drop TABLE Rank_table_blank_2;
create temp table Rank_table_blank_2 as
select
null as rank,
Manager,	Product,	Sale,	Date, sale_rank
from Rank_table
order by Sale DESC ;
-- ранжируем Date по убыванию и заполняем колонку date_rank
--drop TABLE Rank_table_2;
create temp table Rank_table_2 as
select
dense_rank() over (partition by manager order by Date(date) desc) as date_rank,
Manager,	Product,	Sale,	Date, sale_rank
from Rank_table_blank_2
window w as (order by  Date DESC)
order by date_rank;

--выбираем из минимальных значений рангов и группируем
SELECT Manager, Product, Sale, Date from Rank_table_2
where sale_rank =1 and date_rank = 1 group by Manager
```

Комментарии по задачам:
1. В задачи требуется вывести долю, которая приходится на 1 число, тогда как у вас выводится вся информация по контактам на 1 число
2. Информация должна выводится по каждому продукту - в вашей реализации по тем продуктам, у которых их нет, не будет выводится, они затеряются (необходимая доработка: по этим контрактам должно выводится количество 0).
3. Довольно-таки сложно реализовано, можно проще (создавать новые таблицы необходимости нет), но в целом работать будет.

Просьба реализовать задачи в привычном вам инструменте, необязательно на sql, и прислать решение, главное чтобы было правильно.

1)

SQL:

```
select
CONTRACT_ID,
round(sum(case when EVENT_DT
LIKE "01.%" then round(REVENUE) else 0 end) / sum(round(REVENUE)) , 2)
as CONTRACT_SHARE
FROM CONTRACT_REVENUE GROUP BY CONTRACT_ID
```

Python:

```
# Создаем DataFrame
ddt = [datetime.datetime.strptime(date, "%d.%m.%Y")
     for date in ['01.08.2020', '01.09.2020', '01.09.2020', '01.09.2020', '10.09.2020']]
df = pd.DataFrame({'CONTRACT_ID':[101, 101, 102, 103,103],
          'EVENT_DT': ddt,
          'REVENUE': [200, 200, 300, 200, 150 ]})
# Выбираем начало месяца
df['F_day'] = df['EVENT_DT'].astype("str").str.contains("-01")
# функция условия агрегации данных
def calc_share(rev):
   denominator = np.round(rev.REVENUE.sum(), 2)
   res = {"Contract_sum_first": rev[rev['F_day'] == True]['REVENUE'].sum(),
         "Contract_sum": rev['REVENUE'].sum()}
   return pd.Series(res, index=['Contract_sum_first','Contract_sum'])
# группируем
df = df.groupby('CONTRACT_ID').apply(calc_share)
# округляем и выводим доли контрактов на первое число месяца
df['Share_Contract_f_day'] = np.round(df['Contract_sum_first'] / df['Contract_sum'], 2)
df[['Share_Contract_f_day']]
```

2)

SQL:

```
-- создаем объединяющую временную таблицу
drop TABLE ProductContract;
create temp table ProductContract as
select * from (
select * from PRODUCT) t1
left join
(select * from CONTRACTS_PRODUCTS) t2
on t1.PRODUCT_ID=t2.PRODUCT_ID;
-- выбираем только актуальные контракты на текущий момент
select *,
case when END_DT > DATE('now') and START_DT < DATE('now') then 1 else 0 END
as is_work from ProductContract
```

Python:

```
# Создаем DataFrame
START_DT = [datetime.datetime.strptime(date, "%d.%m.%Y")
     for date in ['01.01.2020', '01.07.2020', '01.09.2020', '01.08.2020', '01.01.2020']]
END_DT = [datetime.datetime.strptime(date, "%d.%m.%Y")
     for date in ['01.07.2020','01.09.2020', '31.12.2099', '31.12.2099', '31.12.2099']]
df_CONTRACTS_PRODUCTS = pd.DataFrame({'CONTRACT_ID':['101', '101', '101', '102','103'],
          'PRODUCT_ID': [1, 3, 4, 2, 4],
          'START_DT': START_DT,
           'END_DT': END_DT})
df_PRODUCT = pd.DataFrame({'PRODUCT_ID': [1, 2,3,4,5],
         'PRODUCT_NAME': ['o2 Blue S','o2 Blue M','o2 Free S','o2 Free M','o2 Free L']})
# объединяем обе таблицы и создаем колонку булевыми значениями под актуальные
контракты
today = pd.to_datetime(datetime.date.today())
filter_date = (pd.to_datetime(df_CONTRACTS_PRODUCTS['START_DT']) < today) & \
(pd.to_datetime(df_CONTRACTS_PRODUCTS['END_DT']) > today)
df_CONTRACTS_PRODUCTS['is_works'] = filter_date
df_tmp = pd.merge(df_CONTRACTS_PRODUCTS, df_PRODUCT, on='PRODUCT_ID',
how='outer').fillna(False)
df_tmp2= df_tmp[['CONTRACT_ID','PRODUCT_ID','is_works', 'PRODUCT_NAME']]
# функция условия агрегации данных
def is_available(table):
   res = {'CONTRACTS': table[table['is_works'] == True]['CONTRACT_ID'].count(),
       'CONTRACT_ID': table[table['is_works'] == True]['CONTRACT_ID'].apply(lambda x: x + ',
').sum(),
       'PRODUCT_NAME': table[table['is_works'] == True]['PRODUCT_NAME'].apply(lambda x: x +
', ').sum()}
   return pd.Series(res, index=['CONTRACT_ID','CONTRACTS', 'PRODUCT_NAME'])
```

```
# группируем и применяем функцию
df_res = df_tmp2.groupby('PRODUCT_ID').apply(is_available)
df_res['CONTRACT_ID'] = df_res['CONTRACT_ID'].str.rstrip(' ,')
df_res['PRODUCT_NAME'] = df_res['PRODUCT_NAME'].str.rstrip(' ,')
```

3)
Python:

```
# Создаем DataFrame
ddt = [datetime.datetime.strptime(date, "%d.%m.%Y")
    for date in
    ['10.01.2020', '20.01.2020', '30.12.2019', '10.02.2020', '11.02.2020', '11.02.2020',
'30.12.2019', '10.01.2020']]
df = pd.DataFrame({'Manager':[1, 1, 1, 2, 2, 2, 3, 3],
          'Product': [10, 20, 30, 10, 20, 30, 20, 30],
          'Sale': [500, 300, 500, 400, 500, 500, 500, 300],
          'Date': ddt})

#Сортируем по 2 полям из условия в порядке убывания и оставляем только уникальных
менеджеров (первую встречу в датасете)
df.sort_values(by=['Sale', 'Date'],ascending=False, inplace=True)
df.drop_duplicates(subset=['Manager'], keep='first')
```