Задача 1: Необходимо написать запрос, который позволит понять, идентичны ли данные в двух таблицах. Порядок хранения данных в таблицах значения не имеет.

```
create table t1(a number, b number);

create table t2(a number, b number);
```

Пример данных:

T1:

a	b
1	1
2	2
2	2
3	3
4	4

T2:

a	b
1	1
2	2
3	3
3	3
4	4

Задача 2: Имеется таблица без первичного ключа. Известно, что в таблице имеется задвоение данных. Необходимо удалить дубликаты из таблицы.

```
create table t (a number, b number);
```

Пример данных:

a	b
1	1
2	2
2	2
3	3
3	3
3	3

Требуемый результат:

a	b
1	1
2	2
3	3

Задача 3: Есть таблица с данными в виде дерева. Необходимо написать запрос для получения дерева от корневого узла, узел 5 и все его потомки не должны попасть в результат, нужно вывести для каждого узла имя его родителя, данные отсортировать в порядке возрастания ID с учетом иерархии

```
create table t (id number, pid number, nam varchar2(255));
```

Пример данных:

ID	PID	NAM
1		Корень
2	1	Узел2
3	1	Узел3
4	2	Узел4
5	4	Узел5
6	5	Узел6
7	4	Узел7

Требуемый результат:

ID	PID	NAM	PARENT_NAM
1		Корень	
2	1	Узел2	Корень
4	2	Узел4	Узел2
7	4	Узел7	Узел4
3	1	Узел3	Корень

Задача 4: Имеется таблица курсов валют следующей структуры:

```
create table rates(curr_id number, -- ид валюты
                   date_rate DATE, -- дата курса
                   rate NUMBER -- значение курса
                   )
```

Курс валюты устанавливается не на каждую календарную дату.
Уникальный ключ: curr_id + date_rate.

Напишите запрос, который покажет действующее значение курса заданной валюты на любую заданную календарную дату.

Пример данных:

CURR_ID	DATE_RATE	RATE
1	01.01.2010	30
2	01.01.2010	40
1	02.01.2010	32
1	05.01.2010	33
2	10.01.2010	41
2	15.01.2010	42

Требуемый результат:

Для валюты 1 на 03.01.2010 получить курс 32

Для валюты 2 на 10.01.2010 получить курс 41

Задача 5: Имеется таблица в данными по платежным документам. Необходимо написать запрос, который выведет все документы того типа, которого за все время было по сумме больше всего. Если таких типов несколько, то вывести все такие типы. Для каждой строки результата вывести промежуточную сумму платежей данного типа от самого раннего до текущеиго платежа включительно.

```
create table payments (id number, pay_type NUMBER, pay_date
date, pay_sum number);
```

ID	PAY_TYPE	PAY_DATE	PAY_SUM
1	1	01.01.2012	100
2	1	02.01.2012	200
3	1	03.01.2012	300
4	1	01.02.2012	400
5	1	01.02.2012	500
6	2	01.01.2012	600
7	2	01.02.2012	700
8	2	01.04.2012	800
9	2	01.05.2012	900
10	2	01.06.2012	1000
11	3	10.01.2012	1100
12	3	01.03.2012	1200
13	3	01.05.2012	1300
14	3	05.05.2012	1400
15	3	01.06.2012	1500

Требуемый результат:

ID	PAY_TYPE	PAY_DATE	PAY_SUM	SM
11	3	10.01.2012	1100	1100
12	3	01.03.2012	1200	2300
13	3	01.05.2012	1300	3600
14	3	05.05.2012	1400	5000
15	3	01.06.2012	1500	6500

Задача 6: По таблице из предыдущего примера написать запрос, который выведет данные общей суммой за каждый месяц по типам документов с итогами по каждому типу и общим итогом.

Требуемый результат:

PAY_TYPE	MON	SM
1	01.2012	600
1	02.2012	900
1		1500
2	01.2012	600
2	02.2012	700
2	04.2012	800
2	05.2012	900
2	06.2012	1000
2		4000
3	01.2012	1100
3	03.2012	1200
3	05.2012	2700
3	06.2012	1500
3		6500
		12000

Задача 7: Напишите генератор непрерывного интервала дат с 10.01.2013 по 10.02.2013 в виде запроса.

Требуемый результат:

DT

10.01.2013

11.01.2013

12.01.2013

13.01.2013

14.01.2013

15.01.2013

16.01.2013

17.01.2013

18.01.2013

19.01.2013

20.01.2013

21.01.2013

22.01.2013

23.01.2013

24.01.2013

25.01.2013

26.01.2013

27.01.2013

28.01.2013

29.01.2013

30.01.2013

31.01.2013

01.02.2013

02.02.2013

03.02.2013

04.02.2013

05.02.2013

06.02.2013

07.02.2013

08.02.2013

09.02.2013

10.02.2013