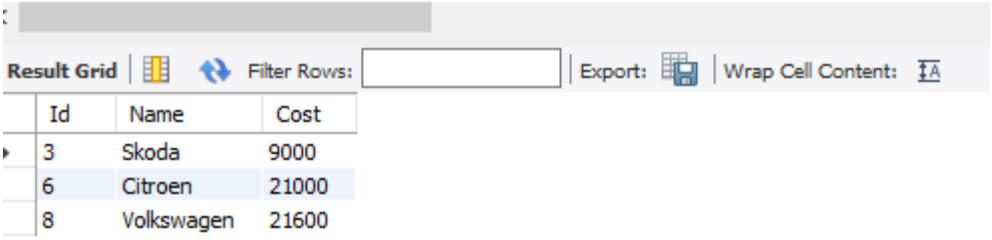
```
# Домашнее задание
Mysql > SELECT * FROM Cars;
+----+------------+--------+
| Id | Name       | Cost   |
+----+------------+--------+
|  1 | Audi       |  52642 |
|  2 | Mercedes   |  57127 |
|  3 | Skoda      |   9000 |
|  4 | Volvo      |  29000 |
|  5 | Bentley    | 350000 |
|  6 | Citroen    |  21000 |
|  7 | Hummer     |  41400 |
|  8 | Volkswagen |  21600 |
+----+------------+--------+
```

1. Создайте представление, в которое попадут автомобили стоимостью до 25 000 долларов

```
~~~
CREATE VIEW CheapCars AS
SELECT * FROM Cars WHERE Cost <= 25000;

SELECT * FROM CheapCars;
~~~
```

```
18  /*
19  1.  Создайте представление,
20  в которое попадут автомобили стоимостью  до 25 000 долларов
21  */
22  CREATE VIEW CheapCars AS
23  SELECT * FROM Cars WHERE Cost <= 25000;
24
25  SELECT * FROM CheapCars;
26
```



Id	Name	Cost
3	Skoda	9000
6	Citroen	21000
8	Volkswagen	21600

2. Изменить в существующем представлении порог для стоимости: пусть цена будет до 30 000 долларов (используя оператор ALTER VIEW)

```
~~~
ALTER VIEW CheapCars AS
SELECT * FROM Cars
WHERE Cost < 30000;

SELECT * FROM CheapCars;
~~~
```

```sql
/*
2.  Изменить в существующем представлении порог для стоимости:
пусть цена будет до 30 000 долларов (используя оператор ALTER VIEW)
*/

ALTER VIEW CheapCars AS
SELECT * FROM Cars
WHERE Cost < 30000;

SELECT * FROM CheapCars;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Id	Name	Cost
3	Skoda	9000
4	Volvo	29000
6	Citroen	21000
8	Volkswagen	21600

3. Создайте представление, в котором будут только автомобили марки “Шкода” и “Ауди”

~~~
CREATE VIEW SkodaAndAudiCars AS
SELECT * FROM Cars
WHERE Name IN ('Skoda', 'Audi');

SELECT * FROM SkodaAndAudiCars;
~~~

```sql
-- 3.Создайте представление, в котором будут только автомобили марки “Шкода” и “Ауди”
CREATE VIEW SkodaAndAudiCars AS
SELECT * FROM Cars
WHERE Name IN ('Skoda', 'Audi');

SELECT * FROM SkodaAndAudiCars;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Id	Name	Cost
1	Audi	52642
3	Skoda	9000

Вывести название и цену для всех анализов, которые продавались 5 февраля 2020 и всю следующую неделю.

~~~
Есть таблица анализов Analysis:
an_id — ID анализа;
an_name — название анализа;
an_cost — себестоимость анализа;
an_price — розничная цена анализа;
an_group — группа анализов.
~~~

~~~
~~~

Есть таблица групп анализов Groups:
gr_id — ID группы;
gr_name — название группы;
gr_temp — температурный режим хранения.
~~~
~~~
Есть таблица заказов Orders:
ord_id — ID заказа;
ord_datetime — дата и время заказа;
ord_an — ID анализа.
~~~

Таблица __Orders__ сформирована с привязкой к таблице __Analysis__ по ключу an_id:
```
INSERT INTO SortedOrders (ord_datetime, ord_an)
SELECT
    CONCAT(DATE_ADD('2020-02-01', INTERVAL FLOOR(RAND() * 15) DAY), ' ',
SEC_TO_TIME(FLOOR(RAND() * 86400))),
    an_id FROM Analysis
WHERE
    DATE_ADD('2020-02-01', INTERVAL FLOOR(RAND() * 15) DAY) BETWEEN
'2020-02-01' AND '2020-02-16';
```
Таблица __SortedOrders__ предназначена для формирования таблицы с последующим ранжированием по дате и заполнением таблицы __Orders__:
~~~
INSERT INTO Orders (ord_datetime, ord_an)
SELECT ord_datetime, ord_an
FROM SortedOrders
ORDER BY ord_datetime;
~~~
Для решения задачи напишем следующий скрипт:
~~~
SELECT an_name, an_price, ord_datetime
FROM Analysis
INNER JOIN Orders ON Analysis.an_id = Orders.ord_an
WHERE ord_datetime BETWEEN '2020-02-05' AND '2020-02-16';
~~~
~~~

```sql
/*
Вывести название и цену для всех анализов,
которые продавались 5 февраля 2020 и всю следующую неделю.
*/

SELECT an_name, an_price, ord_datetime
FROM Analysis
INNER JOIN Orders ON Analysis.an_id = Orders.ord_an
WHERE ord_datetime BETWEEN '2020-02-05' AND '2020-02-16';
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

an_name	an_price	ord_datetime
Computed Tomography	1000	2020-02-07 13:15:18
Thyroid Stimulating Hormone	130	2020-02-07 15:19:32
Mammogram	400	2020-02-08 01:58:54
Chest X-Ray	300	2020-02-08 16:05:24
Urinalysis	100	2020-02-09 04:29:33
Colonoscopy	1500	2020-02-12 00:52:00
Computed Tomography	1000	2020-02-12 08:02:09
Lipid Panel	250	2020-02-12 16:39:18
Electrolyte Panel	180	2020-02-12 22:17:40
Electrolyte Panel	180	2020-02-13 05:51:16
Mammogram	400	2020-02-14 07:49:22
Complete Blood Count	200	2020-02-14 08:06:51
Urinalysis	100	2020-02-15 18:21:03

Добавьте новый столбец под названием «время до следующей станции». Чтобы получить это значение, мы вычитаем время станций для пар смежных станций. Мы можем вычислить это значение без использования оконной функции SQL, но это может быть очень сложно. Проще это сделать с помощью оконной функции LEAD . Эта функция сравнивает значения из одной строки со следующей строкой, чтобы получить результат. В этом случае функция сравнивает значения в столбце «время» для станции со станцией сразу после нее.

train_id integer	station character varying(20)	station_time time without time zone	time_to_next_station interval
110	San Francisco	10:00:00	00:54:00
110	Redwood City	10:54:00	00:08:00
110	Palo Alto	11:02:00	01:33:00
110	San Jose	12:35:00	
120	San Francisco	11:00:00	01:49:00
120	Palo Alto	12:49:00	00:41:00
120	San Jose	13:30:00	

Для решения задачи создадим таблицу __Train__ и заполним её значениями, далее используя скрипт дополним таблицу:

~~~
SELECT *,
SUBTIME (LEAD (station_time, 1) OVER (PARTITION BY train_id ORDER BY train_id), station_time)
AS 'time to next station'
FROM Train;
~~~

```
177 •  SELECT *,
178    SUBTIME (LEAD (station_time, 1) OVER (PARTITION BY train_id ORDER BY train_id), station_time)
179    AS 'time to next station'
180    FROM Train;
181
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

train_id	station	station_time	time to next station
110	San Francisco	10:00:00	00:54:00
110	Redwood City	10:54:00	00:08:00
110	Palo Alto	11:02:00	01:33:00
110	San Jose	12:35:00	NULL
120	San Francisco	11:00:00	01:49:00
120	Palo Alto	12:49:00	00:41:00
120	San Jose	13:30:00	NULL