

SQL assignment:

Aditya Gupta

TAS050

Questions:

1. What is the time period used?
2. How many properties have duplicate entries? Remove duplicate rows (say a row appears 3 times, remove 2 and keep 1)
3. For each property, find out the number of days the property was available and not available (create a table with listing_id, available days, unavailable days and available days as a fraction of total days)
4. How many properties were available on more than 50% of the days? How many properties were available on more than 75% of the days?
5. Create a table with max, min and average price of each property
6. Extract properties with an average price of more than \$500

Solutions

- 1) The format of date is used here as YYYY-MM-DD.**

```
Select * from aditya_data  
Order by date ;  
Select * from aditya_data  
Order by date desc;
```

- 2)** By running the 1st query we get to know there are many rows with the duplicate values. So in the query I write count rows that occur more than 1 .
So to get unique rows i am creating a new table consisting of unique rows.

```
-- checking if there are duplicate rows  
  
select listing_id,date,available,price,count(*)  
from aditya_data  
group by listing_id,date,available,price  
having count(*)>1;  
  
-- creating a new table containing only distinct values  
  
create table new_data as  
select distinct * from aditya_data;  
  
-- total no of enteries in new table  
  
select count(*) from new_data;
```

CODE




listing_id	date	available	price	count(*)
12898806	2017-06-15	f	0	2
12898806	2017-06-14	f	0	2
12898806	2017-06-13	f	0	2
12898806	2017-06-12	f	0	2
12898806	2017-06-11	f	0	2
12898806	2017-06-10	f	0	2
12898806	2017-06-09	f	0	2
12898806	2017-06-08	f	0	2
12898806	2017-06-07	f	0	2
12898806	2017-06-06	f	0	2
12898806	2017-06-05	f	0	2
12898806	2017-06-04	f	0	2
12898806	2017-06-03	f	0	2
12898806	2017-06-02	f	0	2
12898806	2017-06-01	f	0	2
12898806	2017-05-31	f	0	2
12898806	2017-05-30	f	0	2
12898806	2017-05-29	f	0	2
12898806	2017-05-28	f	0	2
12898806	2017-05-27	f	0	2
12898806	2017-05-26	f	0	2
12898806	2017-05-25	f	0	2
12898806	2017-05-24	f	0	2
12898806	2017-05-23	f	0	2
12898806	2017-05-22	f	0	2
12898806	2017-05-21	f	0	2
12898806	2017-05-20	f	0	2

Result 18

it is showing that there are multiple rows with the same data

```
17 select count(*) from new_data;
18
```

100% 31:17

Result Grid   Filter Rows: Export: 

	count(*)
▶ 1305957	

After getting only unique rows we left with 1305957 rows instead of 1308160

- 3) First creating a table name question3 which consist of column listing_id, available days(1 for true and 0 for false) and not available days (1 for false and 0 for true).

Now creating the required table ques3 which consist listing_id, no of available days , no of not available days , and the fraction of available days to total days

```

1 • create table question3 as
2   select listing_id,
3     case available
4       when 't' then 1
5       else 0
6     end is_available,
7     case available
8       when 'f' then 1
9       else 0
10    end is_not_available
11   from new_data;
12
13 • select * from question3;
14
15 • create table ques3 as
16   select listing_id, sum(is_available) as AVAILABLE_DAYS,
17     sum(is_not_available) as NOT_AVAILABLE_DAYS,
18     sum(is_available)/(sum(is_available)+sum(is_not_available)) as FRACTION
19   from question3
20   group by listing_id;
21
22 • select * from ques3;

```

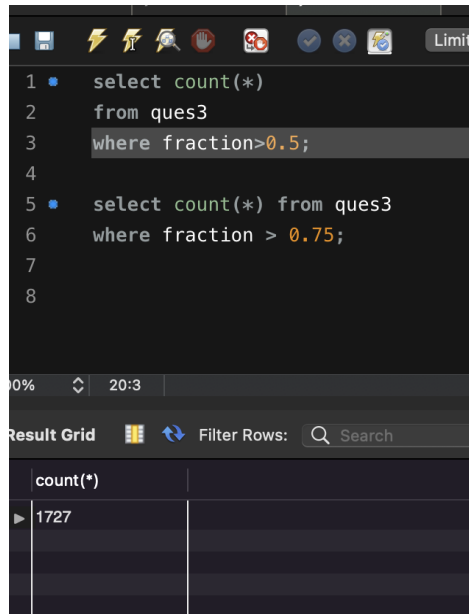
code

	listing_id	AVAILABLE_DAYS	NOT_AVAILABLE_DAYS	FRACTION
	3075044	359	6	0.9836
	6976	319	46	0.8740
	1436513	98	267	0.2685
	7651065	334	31	0.9151
	12386020	58	307	0.1589
	5706985	344	21	0.9425
	2843445	365	0	1.0000
	753446	347	18	0.9507
	849408	107	258	0.2932
	12023024	343	22	0.9397
	1668313	341	24	0.9342
	2684840	0	365	0.0000
	13547301	129	236	0.3534
	5434353	319	46	0.8740
	225979	339	26	0.9288
	3420384	349	16	0.9562
	13512930	0	365	0.0000
	7482195	295	70	0.8082
	7252607	262	103	0.7178
	2583074	332	33	0.9096
	13251243	207	158	0.5671
	225834	299	66	0.8192
	6400432	0	365	0.0000
	5498472	0	365	0.0000
	894539	1	364	0.0027
	879929	122	243	0.3342
	9218312	107	258	0.2932
	321328	356	9	0.9753
	1810172	310	55	0.8493
	6513924	129	236	0.3534
	7093109	353	12	0.9671
	14690527	17	348	0.0466
	7086825	325	40	0.8904
	1861070	362	3	0.9918
	4283698	300	65	0.8219
	10116095	351	14	0.9616
	8548176	351	14	0.9616
	4922204	343	22	0.9397
	4085362	361	4	0.9890
	3755609	339	26	0.9288
	13768853	0	365	0.0000

output

- 4) So from the table we made in previous table we can get the fraction and simply check it for >0.5 and >0.75

code



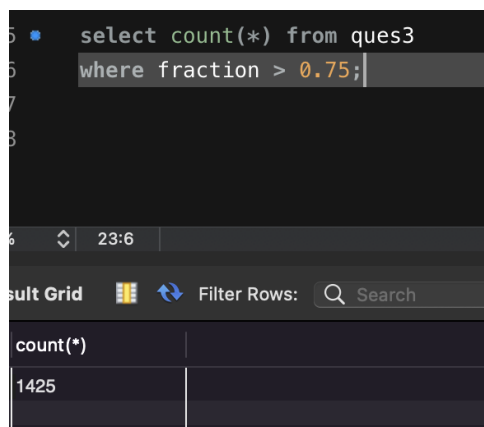
```
1 select count(*)
2 from ques3
3 where fraction>0.5;
4
5 select count(*) from ques3
6 where fraction > 0.75;
7
8
```

0% 20:3

Result Grid Filter Rows: Search

count(*)
1727

Output



```
5 select count(*) from ques3
6 where fraction > 0.75;
```

% 23:6

Result Grid Filter Rows: Search

count(*)
1425

- 5) Creating a table with the columns listing_id, max(price),min(price),avg(price) from table new_data

Code-

```
create table question5 as
select listing_id,
max(price) as MAX_PRICE,
min(nullif(price,0)) as MIN_PRICE,
avg(price) as AVERAGE_PRICE
from new_data
group by listing_id;

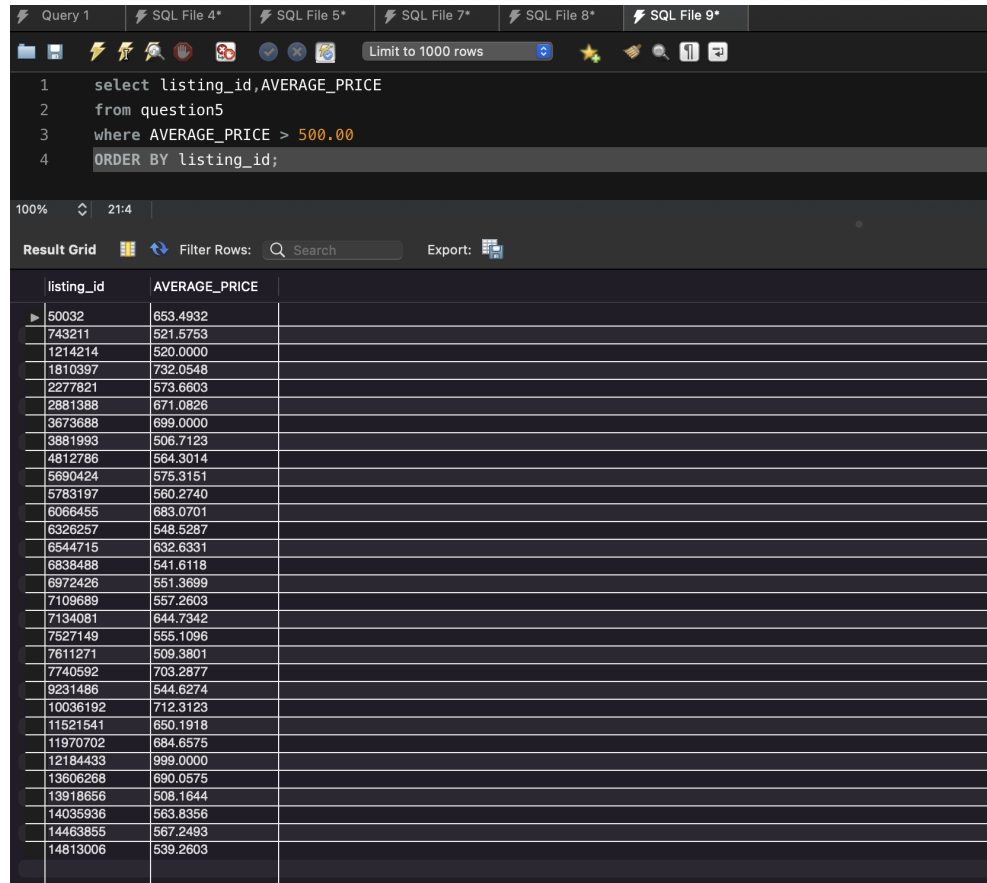
select * from question5 order by listing_id;
```

Output-

	listing_id	MAX_PRICE	MIN_PRICE	AVERAGE_PRICE	
▶	3353	38	32	24.0164	
	5506	275	145	138.7945	
	6695	325	195	175.2329	
	6976	65	65	56.8082	
	8792	154	154	104.6356	
	9273	225	225	224.3836	
	9765	490	192	234.9096	
	9824	490	209	196.7397	
	9855	309	259	265.8247	
	9857	702	301	345.4740	
	9858	699	449	485.4411	
	9860	533	240	260.8904	
	9870	578	240	286.4027	
	9903	299	249	255.8521	
	10730	150	150	38.6301	
	10758	135	115	13.7123	
	10807	145	80	53.7945	
	10809	165	100	12.9726	
	10810	275	250	29.6575	
	10811	175	150	165.5205	
	12356	200	200	149.5890	
	12441	881	399	349.5836	
	13059	688	375	365.6959	
	13589	881	377	380.6521	
	13592	735	299	366.6027	
	18711	385	88	156.8575	
	20000	75	75	70.0685	
	21337	329	329	328.0986	
	21891	399	349	355.5781	
	22208	456	180	228.8603	
	22212	818	285	315.7753	
	22354	135	135	124.2740	
	23619	275	175	122.9041	
	23668	249	199	240.7836	
	24240	250	250	171.9178	
	25142	399	349	355.5781	
	25418	299	249	255.8521	
	27141	578	240	283.4575	
	27611	185	185	37.0000	
	28150	533	240	263.8438	
	29155	881	377	401.1753	
	29765	336	226	206.5425	
	31796	64	60	52.2986	
	36885	155	155	22.5068	
	38579	329	299	302.7014	
	39116	145	100	69.6594	
	question5 1				

6) from the table ques5 getting avg(price) and check if it is >500.

Code and output-



The screenshot shows a SQL IDE interface with a query editor and a results grid. The query editor contains the following SQL code:

```
1 select listing_id,AVERAGE_PRICE
2 from question5
3 where AVERAGE_PRICE > 500.00
4 ORDER BY listing_id;
```

The results grid displays the output of the query, showing a list of listing IDs and their corresponding average prices. The results are sorted by listing ID in ascending order.

listing_id	AVERAGE_PRICE
50032	653.4932
743211	521.5753
1214214	520.0000
1810397	732.0548
2277821	573.6603
2881388	671.0826
3673688	699.0000
3881993	506.7123
4812786	564.3014
5690424	575.3151
5783197	560.2740
6066455	683.0701
6326257	548.5287
6544715	632.6331
6838488	541.6118
6972426	551.3699
7109689	557.2603
7134081	644.7342
7527149	555.1096
7611271	509.3801
7740592	703.2877
9231486	544.6274
10036192	712.3123
11521541	650.1918
11970702	684.6575
12184433	999.0000
13606268	690.0575
13918656	508.1644
14035936	563.8356
14463855	567.2493
14813006	539.2603