In [55]:
```
#importing relevant packages
import json
import pandas as pd
import numpy as np
import pandasql
import datetime
from pandasql import sqldf
from IPython.display import HTML
```

In [56]:
```
# import files
with open('items.json') as json_file:
    menu = json.load(json_file)
with open('orders.json') as json_file:
    orders = json.load(json_file)
```

In [57]:
```
# import data into dataframes
# menus
df_menu = pd.read_json('items.json')
# orders
df_orders = pd.json_normalize(orders,record_path=['items'], meta = ['ordered_at', 'service'
```

In [58]:
```
# check dataframes
df_orders.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70038 entries, 0 to 70037
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   _name           70038 non-null  object
 1   _paid_per_unit  70038 non-null  int64 
 2   _quantity       70038 non-null  int64 
 3   ordered_at      70038 non-null  object
 4   service         70038 non-null  object
 5   name            70038 non-null  object
dtypes: int64(2), object(4)
memory usage: 3.2+ MB
```

In [59]:
```
# check dataframes
df_menu.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 3 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   name            100 non-null    object
 1   cook_time       100 non-null    int64 
 2   price_per_unit  100 non-null    int64 
dtypes: int64(2), object(1)
memory usage: 2.5+ KB
```

In [60]:
```
# convert ordered_at to US Pacific, useful to compute maximum order times
# step 1 - convert to datetime
df_orders['ordered_at'] = pd.to_datetime(df_orders['ordered_at'])
```

In [61]:
```
# step 2 - convert to PST
df_orders['ordered_at'] = df_orders['ordered_at'].dt.tz_localize("UTC")
df_orders['ordered_at_pst'] = df_orders['ordered_at'].dt.tz_convert("US/Pacific")
```

In [62]:
```
# get date
df_orders['ordered_at_date'] = df_orders['ordered_at_pst'].dt.date
# day of the week, useful to get weekly patterns
df_orders['ordered_at_date_day_of_week'] = df_orders['ordered_at_pst'].dt.day_name()
```

In [63]:
```
# check timezone conversion
# check dataframes
df_orders.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70038 entries, 0 to 70037
Data columns (total 9 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   _name                       70038 non-null  object
 1   _paid_per_unit              70038 non-null  int64
 2   _quantity                   70038 non-null  int64
 3   ordered_at                  70038 non-null  datetime64[ns, UTC]
 4   service                     70038 non-null  object
 5   name                        70038 non-null  object
 6   ordered_at_pst              70038 non-null  datetime64[ns, US/Pacific]
 7   ordered_at_date             70038 non-null  object
 8   ordered_at_date_day_of_week  70038 non-null  object
dtypes: datetime64[ns, US/Pacific](1), datetime64[ns, UTC](1), int64(2), object(5)
memory usage: 4.8+ MB
```

In [64]:
```
# review dataset
df_orders
```

Out[64]:

	_name	_paid_per_unit	_quantity	ordered_at	service	name	ordered_at_pst	ordered_at_
0	Cassoulet for Today	262	1	2020-01-20 16:01:00+00:00	SuperEats	Ryan G	2020-01-20 08:01:00-08:00	2020-0
1	Rigatoni with Sausage & Peas	500	1	2020-01-20 16:01:00+00:00	SuperEats	Ryan G	2020-01-20 08:01:00-08:00	2020-0
2	Sugar-Glazed Ham	367	1	2020-01-20 16:04:00+00:00	SuperEats	Amber Hughes	2020-01-20 08:04:00-08:00	2020-0
3	Skillet Ham & Rice	779	1	2020-01-20 16:05:00+00:00	DoorDish	Shawn Brown	2020-01-20 08:05:00-08:00	2020-0
4	Quicker Chicken and Dumplings	349	1	2020-01-20 16:05:00+00:00	DoorDish	Shawn Brown	2020-01-20 08:05:00-08:00	2020-0
...	...	...	...	...	...	...	...	
70033	Skillet Ham & Rice	779	1	2020-01-27 09:57:00+00:00	SuperEats	Crystal Cox	2020-01-27 01:57:00-08:00	2020-0
70034	Seasoned Crab Cakes	786	1	2020-01-27 09:59:00+00:00	SuperEats	Isaac Cooper	2020-01-27 01:59:00-08:00	2020-0
70035	Chicken Ranch Mac & Cheese	779	1	2020-01-27 09:59:00+00:00	SuperEats	Isaac Cooper	2020-01-27 01:59:00-08:00	2020-0

	_name	_paid_per_unit	_quantity	ordered_at	service	name	ordered_at_pst	ordered_at_
70036	The Ultimate Chicken Noodle Soup	249	1	2020-01-27 09:59:00+00:00	SuperEats	Isaac Cooper	2020-01-27 01:59:00-08:00	2020-(
70037	Favorite Chicken Potpie	450	1	2020-01-27 09:59:00+00:00	SuperEats	Isaac Cooper	2020-01-27 01:59:00-08:00	2020-(

70038 rows × 9 columns

In [65]:
```
# checking timezones
df_orders[['ordered_at', 'ordered_at_pst', 'ordered_at_date', 'ordered_at_date_day_of_week'
```

Out[65]:

	ordered_at	ordered_at_pst	ordered_at_date	ordered_at_date_day_of_week
0	2020-01-20 16:01:00+00:00	2020-01-20 08:01:00-08:00	2020-01-20	Monday
1	2020-01-20 16:01:00+00:00	2020-01-20 08:01:00-08:00	2020-01-20	Monday
2	2020-01-20 16:04:00+00:00	2020-01-20 08:04:00-08:00	2020-01-20	Monday
3	2020-01-20 16:05:00+00:00	2020-01-20 08:05:00-08:00	2020-01-20	Monday
4	2020-01-20 16:05:00+00:00	2020-01-20 08:05:00-08:00	2020-01-20	Monday
...	...	...	...	...
70033	2020-01-27 09:57:00+00:00	2020-01-27 01:57:00-08:00	2020-01-27	Monday
70034	2020-01-27 09:59:00+00:00	2020-01-27 01:59:00-08:00	2020-01-27	Monday
70035	2020-01-27 09:59:00+00:00	2020-01-27 01:59:00-08:00	2020-01-27	Monday
70036	2020-01-27 09:59:00+00:00	2020-01-27 01:59:00-08:00	2020-01-27	Monday
70037	2020-01-27 09:59:00+00:00	2020-01-27 01:59:00-08:00	2020-01-27	Monday

70038 rows × 4 columns

In [66]:
```
# importing SQL capabilities
from pandasql import sqldf
mysql = lambda q: sqldf(q, globals())
# tables for dataframes
orders = df_orders
menu = df_menu
```

In [133…
```
order_cnt = '''
SELECT
COUNT(*) FROM orders
WHERE ordered_at_date = '2020-01-25'
AND ordered_at_pst BETWEEN '2020-01-25 20:00:00.000000' AND '2020-01-25 21:00:00.000000'
'''
```

```
order_cnt = mysql(order_cnt)
# neatly print
order_cnt.style
```

Out[133…

	COUNT(*)
0	1275

In [67]:

```
# Total Revenue and usage by day of the week
daily_rev_usage = '''
WITH daily_tots AS
(
SELECT
ordered_at_date,
ordered_at_date_day_of_week,
SUM(_paid_per_unit) AS daily_revenue,
SUM(_quantity) AS daily_orders_cnt
FROM orders
WHERE ordered_at_date NOT IN ('2020-01-27')
GROUP BY 1,2
ORDER BY 1
),

total AS
(
SELECT
SUM(_paid_per_unit) AS total_revenue,
SUM(_quantity) AS total_orders_cnt
FROM orders
WHERE ordered_at_date NOT IN ('2020-01-27')
)

SELECT
A.ordered_at_date,
A.ordered_at_date_day_of_week,
A.daily_revenue,
CAST(A.daily_revenue AS REAL)/CAST(B.total_revenue AS REAL) as pct_revenue,
A.daily_orders_cnt,
CAST(A.daily_orders_cnt AS REAL)/CAST(B.total_orders_cnt AS REAL) AS pct_orders
FROM daily_tots A, total B

'''
daily_rev_usage = mysql(daily_rev_usage)
# neatly print
daily_rev_usage.style
```

Out[67]:

	ordered_at_date	ordered_at_date_day_of_week	daily_revenue	pct_revenue	daily_orders_cnt	pct_orders
0	2020-01-20	Monday	4313115	0.127204	11143	0.127834
1	2020-01-21	Tuesday	4015912	0.118438	10256	0.117658
2	2020-01-22	Wednesday	3399461	0.100258	8693	0.099727
3	2020-01-23	Thursday	3980071	0.117381	10179	0.116775
4	2020-01-24	Friday	5061641	0.149279	13032	0.149504
5	2020-01-25	Saturday	6794296	0.200379	17515	0.200934
6	2020-01-26	Sunday	6342664	0.187060	16350	0.187569

In [68]:

```
# covert to numeric for viz
daily_rev_usage['ordered_at_date'] = pd.to_datetime(daily_rev_usage['ordered_at_date'])
daily_rev_usage['ordered_at_date_day_of_week']=daily_rev_usage['ordered_at_date_day_of_week
```

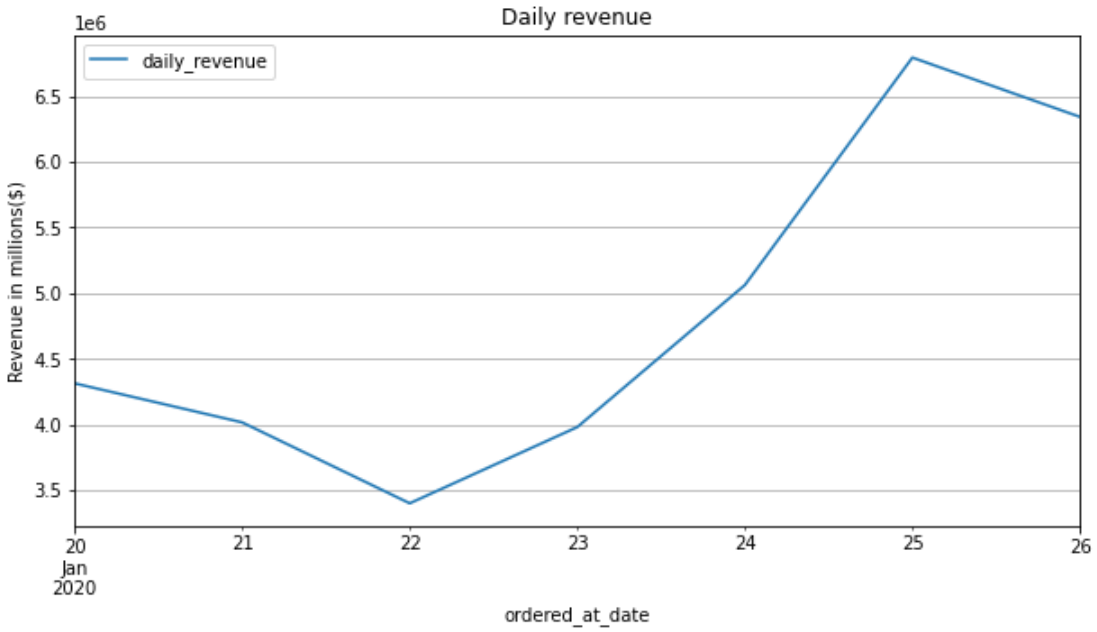
```
daily_rev_usage['daily_revenue']=daily_rev_usage['daily_revenue'].astype(int)
daily_rev_usage['daily_orders_cnt']=daily_rev_usage['daily_orders_cnt'].astype(int)
```

In [69]:

```
# plot results
import matplotlib.pyplot as plt
# Daily Revenue
daily_rev_usage.plot(x ='ordered_at_date', y=['daily_revenue'], kind='line', figsize=(10,5)
             ylabel='Revenue in millions($)')
```

Out[69]:

```
<AxesSubplot:title={'center':'Daily revenue'}, xlabel='ordered_at_date', ylabel='Revenue in
millions($)'>
```

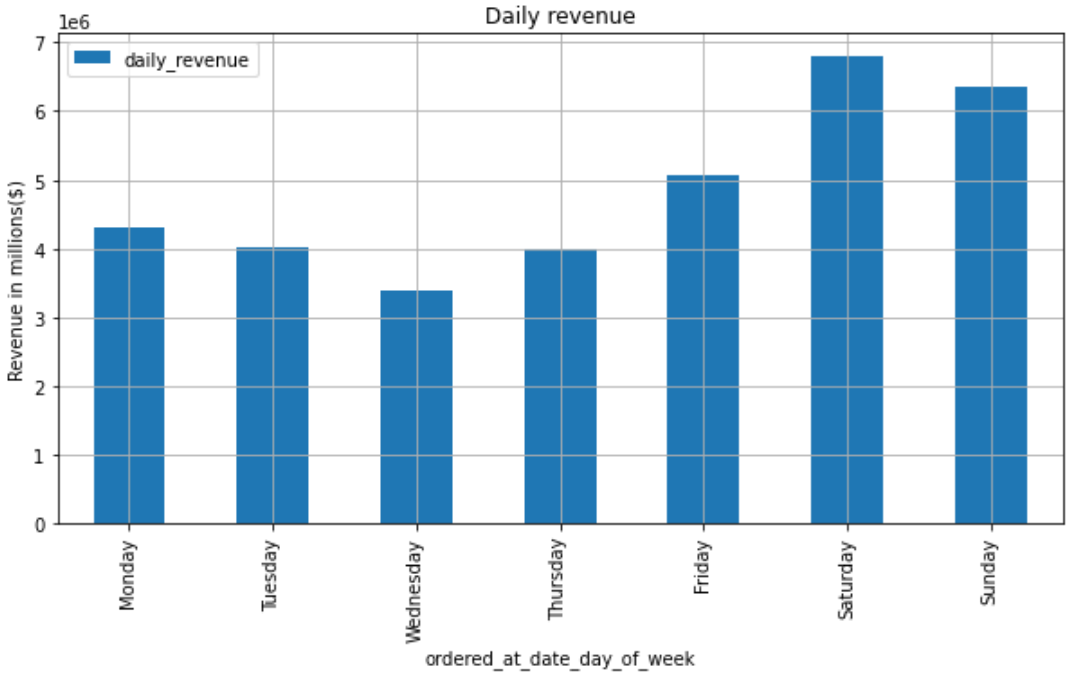


In [70]:

```
# Revenue by day of the week
daily_rev_usage.plot(x ='ordered_at_date_day_of_week', y='daily_revenue', kind='bar', figsi
             ylabel='Revenue in millions($)')
```

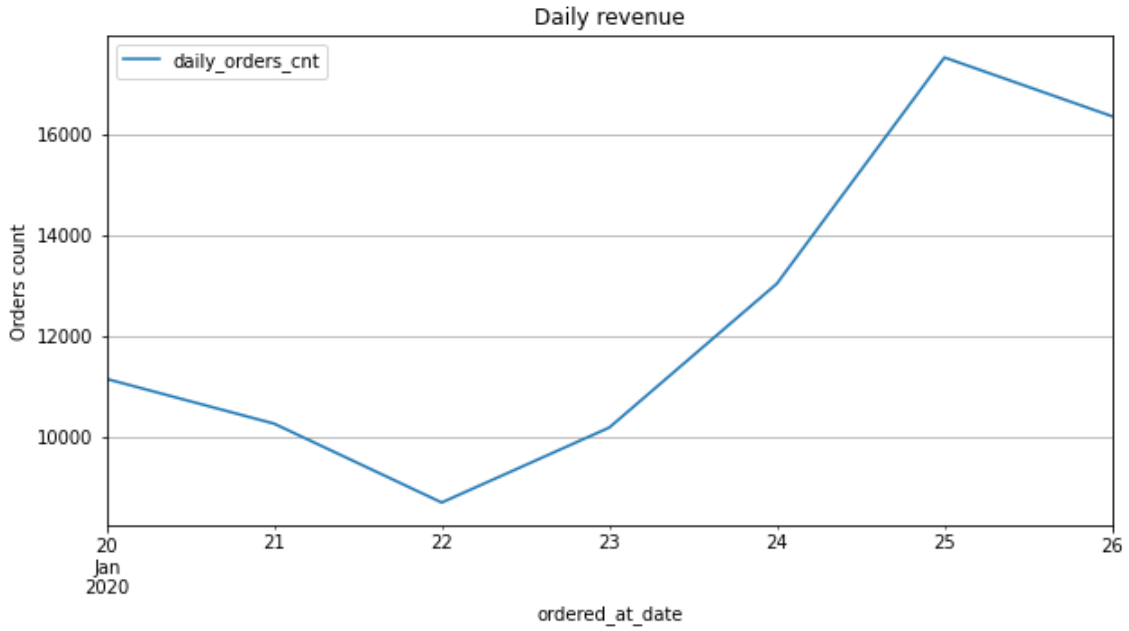
Out[70]:

```
<AxesSubplot:title={'center':'Daily revenue'}, xlabel='ordered_at_date_day_of_week', ylabel
='Revenue in millions($)'>
```



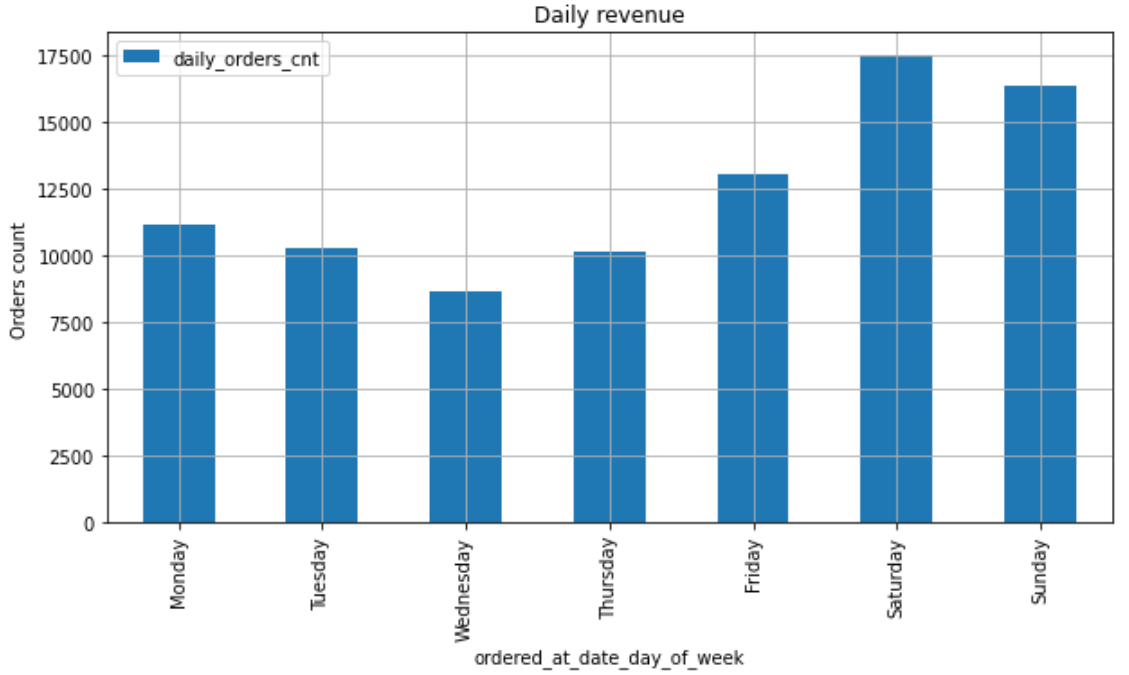
In [71]:
```
# Daily order count
daily_rev_usage.plot(x ='ordered_at_date', y=['daily_orders_cnt'], kind='line', figsize=(10
              ylabel='Orders count')
```

Out[71]:
```
<AxesSubplot:title={'center':'Daily revenue'}, xlabel='ordered_at_date', ylabel='Orders cou
nt'>
```



In [72]:
```
# Order count by the day of the week
daily_rev_usage.plot(x ='ordered_at_date_day_of_week', y=['daily_orders_cnt'], kind='bar',
              ylabel='Orders count')
```

Out[72]:
```
<AxesSubplot:title={'center':'Daily revenue'}, xlabel='ordered_at_date_day_of_week', ylabel
='Orders count'>
```



In [73]:
```
# Saturday and Sundays are the most busy days
# This analysis breaks out order count by hour of the day to find the busiest times of the
order_breakdown_weekends = '''
SELECT
strftime ('%H',ordered_at_pst) hour_of_day,
```

```
SUM(_quantity) AS hourly_orders_cnt
FROM orders
WHERE ordered_at_date IN ('2020-01-25','2020-01-26')
GROUP BY strftime ('%H',ordered_at_pst)
ORDER BY 1
'''

order_breakdown_weekends = mysql(order_breakdown_weekends)
# neatly print
order_breakdown_weekends.style
```

Out[73]:

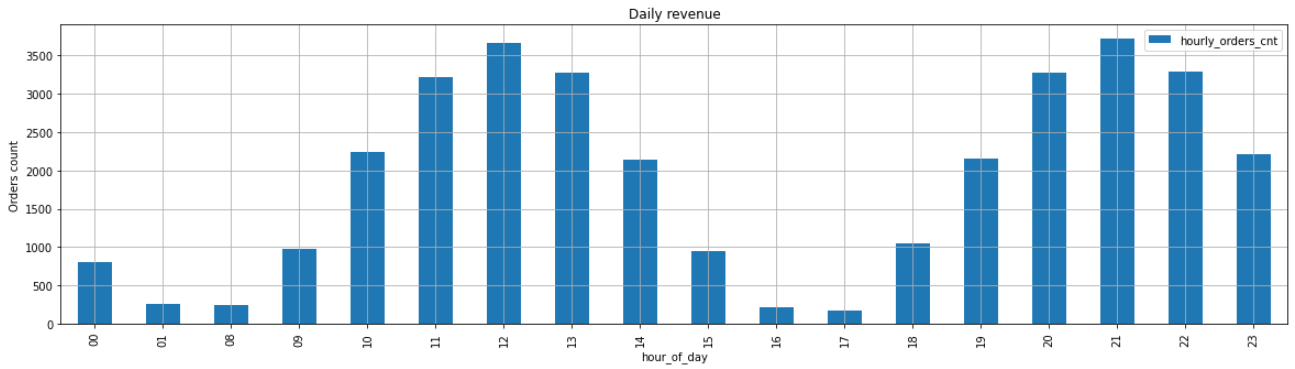
	hour_of_day	hourly_orders_cnt
0	00	811
1	01	257
2	08	244
3	09	978
4	10	2247
5	11	3219
6	12	3655
7	13	3281
8	14	2135
9	15	945
10	16	216
11	17	173
12	18	1055
13	19	2155
14	20	3269
15	21	3715
16	22	3294
17	23	2216

In [74]:

```
# Order count by hour for Saturday and Sunday
order_breakdown_weekends.plot(x ='hour_of_day', y=['hourly_orders_cnt'], kind='bar', figsiz
                      grid=True, legend=True, title='Daily revenue', ylabel='Orders count')
```

Out[74]:

```
<AxesSubplot:title={'center':'Daily revenue'}, xlabel='hour_of_day', ylabel='Orders count'>
```



In [75]:

```
# Orders for weekdays
# This analysis breaks out order count by hour of the day to find the busiest times of the
```

```
order_breakdown_weekends = '''
WITH orders_breakdown AS
(
SELECT
strftime ('%H',ordered_at_pst) hour_of_day,
SUM(_quantity) AS hourly_orders_cnt
FROM orders
WHERE ordered_at_date NOT IN ('2020-01-25','2020-01-26', '2020-01-27')
GROUP BY strftime ('%H',ordered_at_pst)
ORDER BY 1
),

totals AS
(
SELECT
SUM(_quantity) AS hourly_orders_total
FROM orders
WHERE ordered_at_date NOT IN ('2020-01-25','2020-01-26', '2020-01-27')
)

SELECT
A.hour_of_day,
A.hourly_orders_cnt,
B.hourly_orders_total,
ROUND((CAST(A.hourly_orders_cnt AS REAL)/CAST(B.hourly_orders_total AS REAL)),2) AS pct_tot
FROM orders_breakdown A, totals B
'''

order_breakdown_weekends = mysql(order_breakdown_weekends)
# neatly print
order_breakdown_weekends.style
```

Out[75]:

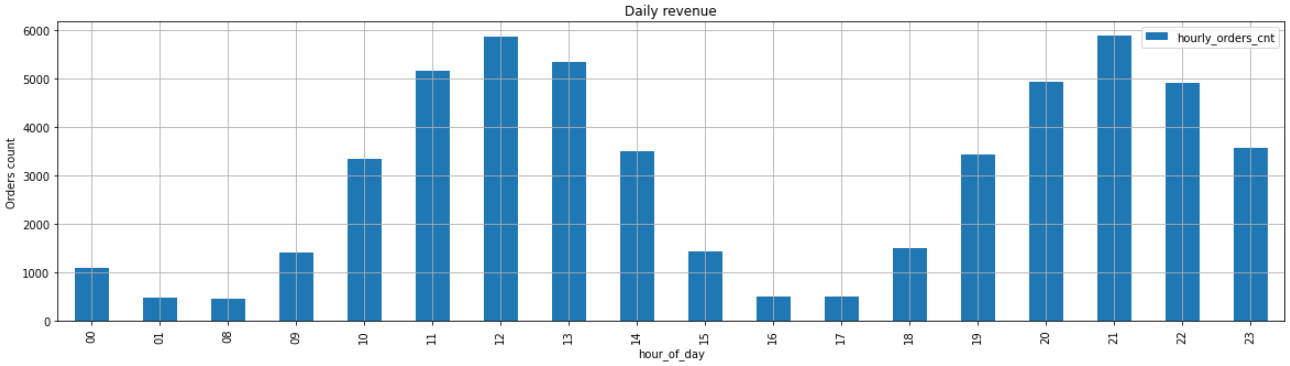
	hour_of_day	hourly_orders_cnt	hourly_orders_total	pct_total
0	00	1103	53303	0.020000
1	01	475	53303	0.010000
2	08	467	53303	0.010000
3	09	1420	53303	0.030000
4	10	3332	53303	0.060000
5	11	5161	53303	0.100000
6	12	5857	53303	0.110000
7	13	5334	53303	0.100000
8	14	3491	53303	0.070000
9	15	1436	53303	0.030000
10	16	513	53303	0.010000
11	17	499	53303	0.010000
12	18	1495	53303	0.030000
13	19	3437	53303	0.060000
14	20	4935	53303	0.090000
15	21	5874	53303	0.110000
16	22	4906	53303	0.090000
17	23	3568	53303	0.070000

In [76]:

```
# Order count by hour for weekdays
```

```
order_breakdown_weekends.plot(x ='hour_of_day', y=['hourly_orders_cnt'], kind='bar', figsiz
                      grid=True, legend=True, title='Daily revenue', ylabel='Orders count')
```

Out[76]: <AxesSubplot:title={'center':'Daily revenue'}, xlabel='hour_of_day', ylabel='Orders count'>



In [77]:

```
# ordered items by hour
ordered_items ='''
SELECT
_name AS item_name,
strftime ('%H',ordered_at_pst) hour_of_day,
SUM(_quantity) AS hourly_orders_cnt
FROM orders
GROUP BY strftime ('%H',ordered_at_pst)
ORDER BY 1,2
'''
ordered_items = mysql(ordered_items)
ordered_items['hour_of_day']=ordered_items['hour_of_day'].astype(int)

# top item during peak hours
top3_items = '''
SELECT
item_name,
hour_of_day,
hourly_orders_cnt
FROM ordered_items
WHERE hour_of_day IN (11,12,13,20,21,22)
ORDER BY 2
'''
top3_items = mysql(top3_items)
top3_items.style

# # covert to numeric for viz
# top3_ordered_items['hour_of_day']=daily_rev_usage['hour_of_day'].astype(IN)
# # neatly print
# top3_ordered_items.style
```

Out[77]:

	item_name	hour_of_day	hourly_orders_cnt
0	Porcini Mac & Cheese	11	8380
1	Chicken Potpie Casserole	12	9512
2	Slow-Cooker Pot Roast	13	8615
3	Frito Pie	20	8204
4	Zucchini Hamburger Pie	21	9589
5	Sage Pork Chops with Cider Pan Gravy	22	8200

In [78]:

```
# Top service provider by orders count
service_providers_cnt = '''
SELECT
```

```
service,
SUM(_quantity) AS orders_cnt
FROM orders
GROUP BY 1
ORDER BY 2 DESC
'''

service_providers_cnt = mysql(service_providers_cnt)
# neatly print
service_providers_cnt.style
```

Out[78]:

	service	orders_cnt
0	SuperEats	52660
1	DoorDish	26116
2	GrubDub	8967

In [79]:

```
# Top service provider by revenue
service_providers_cnt_dollars = '''
SELECT
service,
SUM(_paid_per_unit)/100 AS ordered_total_dollars
FROM orders
GROUP BY 1
ORDER BY 2 DESC
'''

service_providers_cnt_dollars = mysql(service_providers_cnt_dollars)
# neatly print
service_providers_cnt_dollars.style
```

Out[79]:

	service	ordered_total_dollars
0	SuperEats	198325
1	DoorDish	99709
2	GrubDub	43326

In [80]:

```
# Top 10 customers bringin most revenue and their order count
top10_customers = '''
SELECT
name AS customer_name,
SUM(_quantity) AS orders_cnt,
SUM(_paid_per_unit)/100 AS orders_dollar
FROM orders
GROUP BY 1
ORDER BY 3 DESC
LIMIT 10
'''


top10_customers = mysql(top10_customers)
# neatly print
top10_customers.style
```

Out[80]:

	customer_name	orders_cnt	orders_dollar
0	Julie Wright	77	318
1	Brandon Howard	67	295
2	Joseph Smith	77	295

	customer_name	orders_cnt	orders_dollar
3	Taylor Campbell	63	271
4	Joshua Walker	64	266
5	Mary Williams	53	266
6	Kaitlyn Walker	73	263
7	Lauren Wright	60	261
8	Cameron Campbell	52	260
9	Alyssa Scott	57	259

In [121…

```
# top 5 items with most surge premium during lunch and dinner hours
surge_pricing = '''
WITH orders_breakdown AS
(
SELECT
strftime ('%H',A.ordered_at_pst) hour_of_day,
A._name as item_name,
SUM(A._paid_per_unit) AS price_paid,
SUM(B.price_per_unit) AS base_price
FROM orders A
JOIN menu B on A._name = B.name
WHERE ordered_at_date NOT IN ('2020-01-28')
GROUP BY strftime ('%H',ordered_at_pst),2
),

surges AS
(
SELECT
hour_of_day,
item_name,
CASE WHEN hour_of_day IN ('11','12', '13') THEN 'lunch' ELSE 'dinner' END as time_of_day,
(price_paid-base_price) AS surge_premium
FROM orders_breakdown
WHERE hour_of_day IN ('11','12', '13', '20', '21', '22')
),

/* rank function is throwing errors */
lunch_top5 AS
(
SELECT
time_of_day,
item_name,
surge_premium
FROM surges
WHERE time_of_day = 'lunch'
ORDER BY 3 DESC
LIMIT 5
),

dinner_top5 AS
(
SELECT
time_of_day,
item_name,
surge_premium
FROM surges
WHERE time_of_day = 'dinner'
ORDER BY 3 DESC
LIMIT 5
)
```

```
SELECT * FROM lunch_top5
UNION ALL
SELECT * FROM dinner_top5
'''

surge_pricing = mysql(surge_pricing)
# neatly print
surge_pricing.style
```

Out[121…

	time_of_day	item_name	surge_premium
0	lunch	Spaghetti Pie Casserole	1131
1	lunch	Garlic Herbed Beef Tenderloin	1007
2	lunch	Standing Rib Roast	974
3	lunch	Creamy Paprika Pork	891
4	lunch	Skillet Ham & Rice	886
5	dinner	Sage Pork Chops with Cider Pan Gravy	1153
6	dinner	Sage Pork Chops with Cider Pan Gravy	1082
7	dinner	Standing Rib Roast	1026
8	dinner	Chicken Potpie Casserole	985
9	dinner	Skillet Ham & Rice	908

In [149…

```
# trying to find optimum number of cooks in the kitchen
# to simplify the exercise, picking the example of peak hour on a saturday i.e. dinner time
order_breakdown_weekends = '''
WITH cooktimes AS
(
SELECT
A._name AS item_name,
B.cook_time,
SUM(A._quantity) AS ordered_quantity,
ROUND((SUM(A._quantity)*B.cook_time)/60.00,2) AS total_cook_duration_mins
FROM orders A
JOIN menu B ON A._name = B.name
WHERE ordered_at_date = '2020-01-25'
AND ordered_at_pst BETWEEN '2020-01-25 20:00:00.000000' AND '2020-01-25 21:00:00.000000'
GROUP BY 1,2
ORDER BY 4 DESC
)

SELECT
SUM(total_cook_duration_mins),
SUM(ordered_quantity),
SUM(total_cook_duration_mins)/SUM(ordered_quantity) AS avg_duration_per_item_minutes
FROM cooktimes
'''

order_breakdown_weekends = mysql(order_breakdown_weekends)
# neatly print
order_breakdown_weekends.style
```

Out[149…

	SUM(total_cook_duration_mins)	SUM(ordered_quantity)	avg_duration_per_item_minutes
0	14327.000000	1573	9.108074

In [ ]: