



```
%python
from pyspark.sql import functions as F
from pyspark.sql import types as T
from pyspark.sql.functions import col,to_date
# preprocess data

rdd1 =
spark.read.csv('/FileStore/tables/COVID_19_Daily_Counts_of_Cases__Hospitalizations__and_Deaths.csv', header= True, inferSchema= True)
rdd1.printSchema()
```

```
root
|-- DATE_OF_INTEREST: string (nullable = true)
|-- CASE_COUNT: integer (nullable = true)
|-- HOSPITALIZED_COUNT: integer (nullable = true)
|-- DEATH_COUNT: integer (nullable = true)
|-- DEATH_COUNT_PROBABLE      : integer (nullable = true)
|-- CASE_COUNT_7DAY_AVG      : integer (nullable = true)
|-- HOSP_COUNT_7DAY_AVG: integer (nullable = true)
|-- DEATH_COUNT_7DAY_AVG: integer (nullable = true)
|-- BX_CASE_COUNT: integer (nullable = true)
|-- BX_HOSPITALIZED_COUNT: integer (nullable = true)
|-- BX_DEATH_COUNT: integer (nullable = true)
|-- BX_CASE_COUNT_7DAY_AVG: integer (nullable = true)
|-- BX_HOSPITALIZED_COUNT_7DAY_AVG: integer (nullable = true)
|-- BX_DEATH_COUNT_7DAY_AVG: integer (nullable = true)
|-- BK_CASE_COUNT: integer (nullable = true)
|-- BK_HOSPITALIZED_COUNT: integer (nullable = true)
|-- BK_DEATH_COUNT: integer (nullable = true)
|-- BK_CASE_COUNT_7DAY_AVG: integer (nullable = true)
|-- BK_HOSPITALIZED_COUNT_7DAY_AVG: integer (nullable = true)
|-- BK_DEATH_COUNT_7DAY_AVG: integer (nullable = true)
```

```
# only keep date, cases, hospitalized and death
col_covid = ['DATE', 'CASE_COUNT', 'HOSPITALIZED_COUNT', 'DEATH_COUNT']
covid = rdd1.withColumn('DATE', to_date('DATE_OF_INTEREST', 'MM/dd/yyyy'))
).select(col_covid).sort('DATE', ascending=False)
print(covid.count())
covid.show()
covid.write.saveAsTable('COVID19')
```

```
266
```

```
+-----+-----+-----+-----+
```

DATE	CASE_COUNT	HOSPITALIZED_COUNT	DEATH_COUNT
2020-11-20	831	47	4
2020-11-19	1169	88	3
2020-11-18	1282	95	7
2020-11-17	1337	78	9
2020-11-16	1589	110	9
2020-11-15	816	82	12
2020-11-14	947	81	10
2020-11-13	1418	75	6
2020-11-12	1408	65	14
2020-11-11	1436	78	9
2020-11-10	1508	66	7
2020-11-09	1511	73	4
2020-11-08	761	51	9
2020-11-07	798	55	14
2020-11-06	1002	53	10
2020-11-05	1104	71	10

```
# preprocess shooting data
```

```
rdd2 =
```

```
spark.read.csv('/FileStore/tables/NYPD_Shooting_Incident_Data__Year_To_Date_.csv', header = True, inferSchema=True)
```

```
rdd2.printSchema()
```

```
root
```

```
|-- INCIDENT_KEY: integer (nullable = true)
|-- OCCUR_DATE: string (nullable = true)
|-- OCCUR_TIME: string (nullable = true)
|-- BORO: string (nullable = true)
|-- PRECINCT: integer (nullable = true)
|-- JURISDICTION_CODE: integer (nullable = true)
|-- LOCATION_DESC: string (nullable = true)
|-- STATISTICAL_MURDER_FLAG: boolean (nullable = true)
|-- PERP_AGE_GROUP: string (nullable = true)
|-- PERP_SEX: string (nullable = true)
|-- PERP_RACE: string (nullable = true)
|-- VIC_AGE_GROUP: string (nullable = true)
|-- VIC_SEX: string (nullable = true)
|-- VIC_RACE: string (nullable = true)
|-- X_COORD_CD: integer (nullable = true)
|-- Y_COORD_CD: integer (nullable = true)
|-- Latitude: double (nullable = true)
|-- Longitude: double (nullable = true)
|-- New Georeferenced Column: string (nullable = true)
```

```

shooting = rdd2.withColumn('DATE', to_date('OCCUR_DATE', 'MM/dd/yyyy'))
).select(['DATE', 'BORO', 'LOCATION_DESC', 'PERP_AGE_GROUP', 'PERP_SEX',
'PERP_RACE', 'VIC_AGE_GROUP', 'VIC_SEX', 'VIC_RACE',
'STATISTICAL_MURDER_FLAG']).sort('DATE', ascending = False)
shooting.printSchema()
shooting.show()
shooting.write.saveAsTable('SHOOTING')

```

```

root
|-- DATE: date (nullable = true)
|-- BORO: string (nullable = true)
|-- LOCATION_DESC: string (nullable = true)
|-- PERP_AGE_GROUP: string (nullable = true)
|-- PERP_SEX: string (nullable = true)
|-- PERP_RACE: string (nullable = true)
|-- VIC_AGE_GROUP: string (nullable = true)
|-- VIC_SEX: string (nullable = true)
|-- VIC_RACE: string (nullable = true)
|-- STATISTICAL_MURDER_FLAG: boolean (nullable = true)

+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
|    DATE |    BORO | LOCATION_DESC | PERP_AGE_GROUP | PERP_SEX | PERP_RACE | VIC_AGE_GROUP | VIC_SEX | VIC_RACE | STATISTICAL_MURDER_FLAG |
+-----+-----+-----+-----+-----+-----+
| 2020-09-30 | BROOKLYN | MULTI DWELL - APT... | 18-24 | M | BLACK | 18-24 | M | WHITE HISPANIC | false |
| 2020-09-30 | QUEENS | MULTI DWELL - APT... | 25-44 | M | WHITE H |

```

```

%python
# preprocess prisoner data

rdd3 = spark.read.csv('/FileStore/tables/Daily_Inmates_In_Custody.csv', header
= True, inferSchema=True)
rdd3.printSchema()
rdd3.show()

```

```

root
|-- INMATEID: integer (nullable = true)
|-- ADMITTED_DT: string (nullable = true)
|-- DISCHARGED_DT: string (nullable = true)
|-- CUSTODY_LEVEL: string (nullable = true)
|-- BRADH: string (nullable = true)
|-- RACE: string (nullable = true)
|-- GENDER: string (nullable = true)

```

```

|-- AGE: integer (nullable = true)
|-- INMATE_STATUS_CODE: string (nullable = true)
|-- SEALED: string (nullable = true)
|-- SRG_FLG: string (nullable = true)
|-- TOP_CHARGE: string (nullable = true)
|-- INFRACTION: string (nullable = true)

+-----+-----+-----+-----+-----+-----+
--+-----+-----+-----+-----+-----+
| INMATEID|          ADMITTED_DT|DISCHARGED_DT|CUSTODY_LEVEL|BRADH|RACE|GENDER|AGE|INMATE_STATUS_CODE|SEALED|SRG_FLG|TOP_CHARGE|INFRACTION|
+-----+-----+-----+-----+-----+-----+

```

```
%python
```

```

string_udf = F.udf(lambda x: x[:10])
prisoner = rdd3.withColumn('DATE', F.to_date(string_udf('ADMITTED_DT'),
'MM/dd/yyyy'))).select(['DATE', 'CUSTODY_LEVEL', 'BRADH', 'GENDER', 'AGE',
'RACE', 'INFRACTION']).sort('DATE', ascending = False)
prisoner.show()
prisoner.write.saveAsTable('PRISONER')

```

```

+-----+-----+-----+-----+-----+-----+
|          DATE|CUSTODY_LEVEL|BRADH|GENDER|AGE|RACE|INFRACTION|
+-----+-----+-----+-----+-----+-----+
|2020-11-23|      null|    N|    M|  58|   O|          N|
|2020-11-23|      null|    N|    M|  33|   B|          N|
|2020-11-23|      null|    N|    M|  30|   B|          N|
|2020-11-23|      null|    N|    M|  32|   B|          N|
|2020-11-23|      null|    N|    M|  37|   B|          N|
|2020-11-23|      null|    N|    M|  43|   B|          N|
|2020-11-23|      null|    N|    M|  36|   B|          N|
|2020-11-23|      null|    N|    M|  43|   B|          N|
|2020-11-22|      null|    N|    M|  31|   W|          N|
|2020-11-22|      null|    N|    M|  37|   B|          N|
|2020-11-22|      null|    N|    M|  26|   B|          N|
|2020-11-22|      null|    N|    M|  39|   B|          N|
|2020-11-22|      null|    N|    M|  26|   O|          N|
|2020-11-22|      null|    N|    M|  26|   B|          N|
|2020-11-22|      null|    N|    M|  21|   B|          N|
|2020-11-22|      MAX|    N|    M|  28|   B|          N|
|2020-11-22|      null|    N|    M|  27|   B|          N|
|2020-11-22|      null|    N|    M|  31|   B|          N|

```

```
# process utility data
rdd4 =
spark.read.csv('/FileStore/tables/Energy_and_Water_Data_Disclosure_for_Local_La
w_84_2020__Data_for_Calendar_Year_2019_.csv', header=True, inferSchema=True )
cols = rdd4.columns
index = [60, 14, 17, 27, 48, 51, 54, 57]
columns = [cols[i] for i in index]
rdd4 = rdd4.select(columns)
rdd4.columns
```

```
Out[2]: ['Generation Date',
'Borough',
'Primary Property Type - Self Selected',
'Occupancy',
'Weather Normalized Site Natural Gas Use (therms)',
'Weather Normalized Site Electricity (kWh)',
'Total GHG Emissions (Metric Tons CO2e)',
'Water Use (All Water Sources) (kgal)']
```

```
# select columns and convert date
df = rdd4.select(col('Generation Date').alias('Date'), col('Primary Property
Type - Self Selected').alias('Usage'), 'Occupancy', col('Weather Normalized
Site Natural Gas Use (therms)').alias('Gas'), col('Weather Normalized Site
Electricity (kWh)').alias('Electricity'), col('Total GHG Emissions (Metric Tons
CO2e)').alias('Emission'), col('Water Use (All Water Sources)
(kgal)').alias('Water'))
df = df.withColumn('DATE', to_date('Date', 'MM/dd/yyyy') ).sort('DATE',
ascending = False)
df.printSchema()
df.show()
```

```
root
|-- DATE: date (nullable = true)
|-- Usage: string (nullable = true)
|-- Occupancy: integer (nullable = true)
|-- Gas: string (nullable = true)
|-- Electricity: string (nullable = true)
|-- Emission: string (nullable = true)
|-- Water: string (nullable = true)

+-----+-----+-----+-----+-----+-----+
|      DATE|      Usage|Occupancy|      Gas| Electricity| Em
ission|      Water|
+-----+-----+-----+-----+-----+-----+
-----+-----+
```

2020-11-09	Distribution Center	100	3070.5	208908.2
73.1	478.8			
2020-11-09	Multifamily Housing	50	2403.5	Not Available Not Ava
ilable	503.4			
2020-11-09	Medical Office	100	5719.4	505838

```
# clean data and format it as one person usage
```

```
def isfloat(x):
```

```
    try:
```

```
        float(x)
```

```
    return True
```

```
except:
```

```
    return False
```

```
def filter_value(x):
```

```
    if isfloat(x[2]) and x[2] != 0 and isfloat(x[3]) and isfloat(x[4]) and  
isfloat(x[5]) and isfloat(x[6]):
```

```
        return x
```

```
def get_ave(x):
```

```
    return (x[0], x[1], x[2], round(float(x[3]) / float(x[2]), 2),  
round(float(x[4]) / float(x[2]), 2), round(float(x[5]) / x[2], 2),  
round(float(x[6]) / x[2], 2) )
```

```
rdd_df = df.rdd
```

```
print(rdd_df.count())
```

```
header = rdd_df.first()
```

```
rdd_df = rdd_df.subtract(sc.parallelize([header])).map(tuple)
```

```
rdd_df = rdd_df.filter(filter_value)
```

```
print(rdd_df.count())
```

```
rdd_df = rdd_df.map(lambda x: get_ave(x) )
```

```
rdd_df.take(10)
```

```
utility_df = spark.createDataFrame(rdd_df, ['DATE', 'Usage', 'Occupancy',  
'Gas', 'Electricity', 'Emission', 'Water'])
```

```
#utility_df.show()
```

```
utility_df.printSchema()
```

```
utility_df.show()
```

```
utility_df.coalesce(1).write.saveAsTable('UTILITY')
```

```
28807
```

```
18256
```

```
root
```

```
|-- DATE: date (nullable = true)
```

```
|-- Usage: string (nullable = true)
```

```
|-- Occupancy: long (nullable = true)
```

```

|-- Gas: double (nullable = true)
|-- Electricity: double (nullable = true)
|-- Emission: double (nullable = true)
|-- Water: double (nullable = true)

+-----+-----+-----+-----+-----+-----+-----+
-+
|      DATE|      Usage|Occupancy|      Gas|Electricity|Emission|  Water|
+-----+-----+-----+-----+-----+-----+-----+
-+
|2020-11-03| Multifamily Housing|      95| 254.76|      3906.83|      2.45|  39.15|
|2020-11-02|           Hotel|     100| 398.39|     10395.76|      4.93|  41.46|

```

process complaints data which includes conflicts and crimes

```

rdd5 =
spark.read.csv('/FileStore/tables/NYPD_Complaint_Data_Current__Year_To_Date_.csv', header=True, inferSchema=True )
rdd5.printSchema()
rdd5.show()

```

```

root
|-- CMPLNT_NUM: integer (nullable = true)
|-- ADDR_PCT_CD: integer (nullable = true)
|-- BORO_NM: string (nullable = true)
|-- CMPLNT_FR_DT: string (nullable = true)
|-- CMPLNT_FR_TM: string (nullable = true)
|-- CMPLNT_TO_DT: string (nullable = true)
|-- CMPLNT_TO_TM: string (nullable = true)
|-- CRM_ATPT_CPTD_CD: string (nullable = true)
|-- HADEVELOPT: string (nullable = true)
|-- HOUSING_PSA: integer (nullable = true)
|-- JURISDICTION_CODE: integer (nullable = true)
|-- JURIS_DESC: string (nullable = true)
|-- KY_CD: integer (nullable = true)
|-- LAW_CAT_CD: string (nullable = true)
|-- LOC_OF_OCCUR_DESC: string (nullable = true)
|-- OFNS_DESC: string (nullable = true)
|-- PARKS_NM: string (nullable = true)
|-- PATROL_BORO: string (nullable = true)
|-- PD_CD: integer (nullable = true)
|-- PD_DESC: string (nullable = true)

```

```

cols = ['DATE', 'OFNS_DESC', 'SUSP_AGE_GROUP', 'SUSP_RACE', 'SUSP_SEX',
'VIC_AGE_GROUP', 'VIC_RACE', 'VIC_SEX']
rdd5 = rdd5.withColumn('DATE', to_date('Cmplnt_Fr_Dt', 'MM/dd/yyyy'))
).select(cols).sort('DATE', ascending = False)
rdd5.show()
# rdd5.write.saveAsTable('CRIME')
rdd5.printSchema()
# filter imcomplete rows
file_read = rdd5.filter((rdd5['SUSP_AGE_GROUP'] != 'UNKNOWN') &
(rdd5['SUSP_RACE'] != 'UNKNOWN') & (rdd5['SUSP_SEX'] != 'UNKNOWN') &
(rdd5['VIC_AGE_GROUP'] != 'UNKNOWN') & (rdd5['VIC_RACE'] != 'UNKNOWN') &
(rdd5['SUSP_SEX'] != 'UNKNOWN') )
# a problem is the data too big to be saved to a table, so I saved it to a csv
file and then read the file into a table
file_read.write.csv('crime.csv', header = True)

```

```

+-----+-----+-----+-----+-----+-----+
|      DATE|      OFNS_DESC|SUSP_AGE_GROUP|      SUSP_RACE|SUSP_SEX|VIC_AGE_GROUP|      VIC_RACE|VIC_SEX|
+-----+-----+-----+-----+-----+-----+
|2020-09-30|CRIMINAL MISCHIEF...|      UNKNOWN|      UNKNOWN|      U|      25-44|WHITE HISPANIC|      F|
|2020-09-30|      FELONY ASSAULT|      25-44|      BLACK|      F|      UNKNOWN|      UNKNOWN|      E|
|2020-09-30|ASSAULT 3 & RELAT...|      25-44|      BLACK|      M|      25-44|      BLACK|      F|
|2020-09-30|      PETIT LARCENY|      null|      null|      null|      UNKNOWN|      UNKNOWN|      D|
|2020-09-30|      BURGLARY|      UNKNOWN|BLACK HISPANIC|      M|      UNKNOWN|BLACK HISPANIC|      D|
|2020-09-30|ASSAULT 3 & RELAT...|      UNKNOWN|      BLACK|      F|      45-64|WHITE HISPANIC|      F|
|2020-09-30|      GRAND LARCENY|      UNKNOWN|      UNKNOWN|      M|      25-44|      WHITE|      F|
|2020-09-30|ASSAULT 3 & RELAT...|      UNKNOWN|      BLACK|      M|

```

```
%fs rm -r '/crime.csv'
```

```
res19: Boolean = true
```


%sql

```
create table if not exists CRIME
using csv
options(
path "/crime.csv",
inferSchema 'true',
header 'true'
)
```

OK

```
%sql
--- Crime table finished
select * from crime limit 10
```

	DATE ▲	OFNS_DESC ▲	SUSP_AGE_GROUP ▲	SUSP_RA
1	2020-09-30	ASSAULT 3 & RELATED OFFENSES	25-44	WHITE HI
2	2020-09-30	ROBBERY	18-24	BLACK
3	2020-09-30	ASSAULT 3 & RELATED OFFENSES	25-44	BLACK
4	2020-09-30	HARRASSMENT 2	25-44	BLACK H
5	2020-09-30	FELONY ASSAULT	25-44	BLACK
6	2020-09-30	HARRASSMENT 2	18-24	BLACK
7	2020-09-30	ROBBERY	18-24	BLACK

Showing all 10 rows.



```
rdd6 =
spark.read.csv('/FileStore/tables/Motor_Vehicle_Collisions___Crashes.csv',
inferSchema=True, header= True)
rdd6.printSchema()
rdd6.show()
```

```
root
|-- CRASH DATE: string (nullable = true)
|-- CRASH TIME: string (nullable = true)
|-- BOROUGH: string (nullable = true)
|-- ZIP CODE: string (nullable = true)
|-- LATITUDE: double (nullable = true)
|-- LONGITUDE: double (nullable = true)
|-- LOCATION: string (nullable = true)
```

```

|-- ON STREET NAME: string (nullable = true)
|-- CROSS STREET NAME: string (nullable = true)
|-- OFF STREET NAME: string (nullable = true)
|-- NUMBER OF PERSONS INJURED: string (nullable = true)
|-- NUMBER OF PERSONS KILLED: integer (nullable = true)
|-- NUMBER OF PEDESTRIANS INJURED: integer (nullable = true)
|-- NUMBER OF PEDESTRIANS KILLED: integer (nullable = true)
|-- NUMBER OF CYCLIST INJURED: integer (nullable = true)
|-- NUMBER OF CYCLIST KILLED: string (nullable = true)
|-- NUMBER OF MOTORIST INJURED: string (nullable = true)
|-- NUMBER OF MOTORIST KILLED: integer (nullable = true)
|-- CONTRIBUTING FACTOR VEHICLE 1: string (nullable = true)

```

```

incident = rdd6.groupBy('CRASH DATE', 'BOROUGH').agg(F.count('NUMBER OF PERSONS
INJURED').alias('injured_count'), F.count('NUMBER OF PERSONS
KILLED').alias('death_count')).withColumn('Date', to_date('CRASH DATE',
'MM/dd/yyyy')).select(['DATE', 'BOROUGH', 'injured_count', 'death_count']
).sort('DATE', ascending = False)
incident.show()
incident.printSchema()

```

DATE	BOROUGH	injured_count	death_count
2020-11-20	BROOKLYN	70	70
2020-11-20	MANHATTAN	20	20
2020-11-20	STATEN ISLAND	3	3
2020-11-20	QUEENS	53	53
2020-11-20	BRONX	29	29
2020-11-20	null	87	87
2020-11-19	QUEENS	42	42
2020-11-19	MANHATTAN	12	12
2020-11-19	STATEN ISLAND	3	3
2020-11-19	BROOKLYN	52	52
2020-11-19	BRONX	32	32
2020-11-19	null	84	84
2020-11-18	MANHATTAN	23	23
2020-11-18	null	84	84
2020-11-18	BRONX	36	36
2020-11-18	QUEENS	44	44
2020-11-18	BROOKLYN	67	67
2020-11-18	STATEN ISLAND	3	3

```

#print(incident.count())
#incident.write.saveAsTable('Vehicle_Collisions')
incident.select('injured_count')

```

```
%sql
```

```
--- check the vehicle_collision table
```

```
--- select * from vehicle_collisions limit 5;
```

```
select PERP_RACE, count(*) from shooting group by PERP_RACE
```

	PERP_RACE ▲	count(1) ▲
1	WHITE	15
2	BLACK	390
3	null	906
4	BLACK HISPANIC	61
5	WHITE HISPANIC	103
6	UNKNOWN	20
7	ASIAN / PACIFIC ISLANDER	6

Showing all 7 rows.



```
%sql
```

```
CREATE VIEW covid19_view AS
```

```
select year(date) as year, month(date) as month, DAY(date) as day, CASE_COUNT,  
HOSPITALIZED_COUNT, DEATH_COUNT from covid19 order by month desc, day desc
```

OK

```
%sql
```

```
--- create a view with daily prisoners
```

```
CREATE VIEW PRISONER_BY_DAY AS
```

```
select year, month, day, sum(prisoner_count) as prisoner_sum from  
(select month(date) as month, year(date) as year, DAY(date) as day, count(*)  
prisoner_count from prisoner where year(date) = 2020 group by date order by  
prisoner_count desc)  
group by year, month, day order by year desc, month desc , day desc
```

OK

```
%sql
```

```
--- check the view
```

```
select * from prisoner_by_day limit 20
```

	year ▲	month ▲	day ▲	prisoner_sum ▲
--	--------	---------	-------	----------------

1	2020	11	23	8
2	2020	11	22	29
3	2020	11	21	39
4	2020	11	20	51
5	2020	11	19	59
6	2020	11	18	33
7	2020	11	17	34

Showing all 20 rows.



%sql

```
-- get daily shooting
CREATE VIEW DAILY_SHOOTING_VIEW AS
select year(date) as year, month(date) as month, day(date) as day, count(*) as
shooting_case_count from shooting group by year, month, day order by month
desc, day desc
```

	year ▲	month ▲	day ▲	shooting_case_count ▲
1	2020	9	30	8
2	2020	9	29	2
3	2020	9	28	4
4	2020	9	27	8
5	2020	9	26	3
6	2020	9	25	5
7	2020	9	24	4

Showing all 254 rows.



```
%sql
```

```
--- get average daily average utility usage
```

```
CREATE VIEW bridge_table as
```

```
with temp_table(date, gas_ave, electricity_ave, water_ave, emission_ave) as (  
select date, sum(Gas)/ count(*) as gas_ave, sum(Electricity) / count(*) as  
electricity_ave, sum(Water)/ count(*) as water_ave, sum(Emission) / count(*) as  
emission_ave  
from utility  
group by date order by date desc)
```

```
select min(gas_ave) as min_gas, max(gas_ave) as max_gas, min(electricity_ave)  
as min_electricity, max(electricity_ave) as max_electricity, min(water_ave) as  
min_water, max(water_ave) as max_water, min(emission_ave) as min_emission,  
max(emission_ave) as max_emission from temp_table
```

Error in SQL statement: AnalysisException: View `default`.`bridge_table` already exists. If you want to update the view definition, please use ALTER VIEW AS or CREATE OR REPLACE VIEW AS;

```
%sql
```

```
--- get normalize those data and scale up to make them comparable
```

```
CREATE VIEW NORMALIZED_UTILITY AS
```

```
with temp_table(date, gas_ave, electricity_ave, water_ave, emission_ave) as (  
select date, round(sum(Gas)/ count(*), 2) as gas_ave, round(sum(Electricity) /  
count(*), 2) as electricity_ave, round(sum(Water)/ count(*), 2) as water_ave,  
sum(Emission) / count(*) as emission_ave  
from utility  
group by date order by date desc)
```

```
select year(date) as year, month(date) as month, day(date) as day, round(  
(gas_ave - min_gas) / (max_gas - min_gas) * 10000) as norm_gas, round(  
(electricity_ave - min_electricity) / (max_electricity - min_electricity) *  
10000) as norm_electricity, round( (water_ave - min_water) / (max_water -  
min_water) * 10000) as norm_water, round( (emission_ave - min_emission) /  
(max_emission - min_emission) * 10000) as norm_emission from temp_table,  
bridge_table order by year desc, month desc, day desc
```

```
OK
```

```
%sql
```

```
select month, sum(norm_gas) as gas, sum(norm_electricity) as electricity,
sum(norm_water) water, sum(norm_emission) emission from NORMALIZED_UTILITY
group by month order by month desc
```

	month ▲	gas ▲	electricity ▲	water ▲	emission ▲	
1	11	2070	4722	3	2644	
2	10	14437	17930	43	14823	
3	9	25863	22090	10	21403	
4	8	17452	26465	35	20620	
5	7	11332	15380	19	12728	
6	6	10607	13864	14	11562	
7	5	11353	19906	25	24608	

Showing all 9 rows.



```
%sql
```

```
--- create a crime view
```

```
CREATE VIEW CRIME_VIEW AS
```

```
select year(date) as year, month(date) as month, day(date) as day, count(*) as
cases from crime where date > '2019-12-31' group by year, month, day order by
year desc, month desc, day desc
```

OK

Error in SQL statement: AnalysisException: View `default`.`VEHICLE_COLLISIONS_VIEW` already exists. If you want to update the view definition, please use ALTER VIEW AS or CREATE OR REPLACE VIEW AS;