```
import pandas as pd
import numpy as np
```

```
# uploading data file
from google.colab import files
uploader = files.upload()
```

Choose Files uber_data.csv

- **uber_data.csv**(text/csv) - 15824375 bytes, last modified: 7/31/2025 - 100% done

```
Saving uber_data.csv to uber_data.csv
```

```
# reading csv data file
data = pd.read_csv(r"/content/uber_data.csv")  # or just "content/uber_data.csv"
data.head()
```

| | VendorID | tpep_pickup_datetime | tpep_dropoff_datetime | passenger_count | trip_distance | pickup_longitude | pickup_latitude | Ratecod |
|---|---|---|---|---|---|---|---|---|
| **0** | 1 | 2016-03-01 00:00:00 | 2016-03-01 00:07:55 | 1 | 2.50 | -73.976746 | 40.765152 | |
| **1** | 1 | 2016-03-01 00:00:00 | 2016-03-01 00:11:06 | 1 | 2.90 | -73.983482 | 40.767925 | |
| **2** | 2 | 2016-03-01 00:00:00 | 2016-03-01 00:31:06 | 2 | 19.98 | -73.782021 | 40.644810 | |
| **3** | 2 | 2016-03-01 00:00:00 | 2016-03-01 00:00:00 | 3 | 10.78 | -73.863419 | 40.769814 | |
| **4** | 2 | 2016-03-01 00:00:00 | 2016-03-01 00:00:00 | 5 | 30.43 | -73.971741 | 40.792183 | |

Next steps: Generate code with data   View recommended plots   New interactive sheet

```
data['trip_id'] = data.index
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 20 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   VendorID               100000 non-null  int64  
 1   tpep_pickup_datetime   100000 non-null  object 
 2   tpep_dropoff_datetime  100000 non-null  object 
 3   passenger_count        100000 non-null  int64  
 4   trip_distance          100000 non-null  float64
 5   pickup_longitude       100000 non-null  float64
 6   pickup_latitude        100000 non-null  float64
 7   RatecodeID             100000 non-null  int64  
 8   store_and_fwd_flag     100000 non-null  object 
 9   dropoff_longitude      100000 non-null  float64
 10  dropoff_latitude       100000 non-null  float64
 11  payment_type           100000 non-null  int64  
 12  fare_amount            100000 non-null  float64
 13  extra                  100000 non-null  float64
 14  mta_tax                100000 non-null  float64
 15  tip_amount             100000 non-null  float64
 16  tolls_amount           100000 non-null  float64
 17  improvement_surcharge  100000 non-null  float64
 18  total_amount           100000 non-null  float64
 19  trip_id                100000 non-null  int64  
dtypes: float64(12), int64(5), object(3)
memory usage: 15.3+ MB
```

```
# type conversions
data['tpep_pickup_datetime'] = pd.to_datetime(data['tpep_pickup_datetime'])
data['tpep_dropoff_datetime'] = pd.to_datetime(data['tpep_dropoff_datetime'])
```

```
data['tpep_dropoff_datetime']
```

| | tpep_dropoff_datetime |
|---|---|
| **0** | 2016-03-01 00:07:55 |
| **1** | 2016-03-01 00:11:06 |
| **2** | 2016-03-01 00:31:06 |
| **3** | 2016-03-01 00:00:00 |
| **4** | 2016-03-01 00:00:00 |
| **...** | ... |
| **99995** | 2016-03-01 06:22:15 |
| **99996** | 2016-03-01 06:32:41 |
| **99997** | 2016-03-01 06:37:23 |
| **99998** | 2016-03-01 06:22:09 |
| **99999** | 2016-03-01 06:22:00 |

100000 rows × 1 columns

**dtype:** datetime64[ns]

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 20 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   VendorID               100000 non-null  int64
 1   tpep_pickup_datetime   100000 non-null  datetime64[ns]
 2   tpep_dropoff_datetime  100000 non-null  datetime64[ns]
 3   passenger_count        100000 non-null  int64
 4   trip_distance          100000 non-null  float64
 5   pickup_longitude       100000 non-null  float64
 6   pickup_latitude        100000 non-null  float64
 7   RatecodeID             100000 non-null  int64
 8   store_and_fwd_flag     100000 non-null  object
 9   dropoff_longitude      100000 non-null  float64
 10  dropoff_latitude       100000 non-null  float64
 11  payment_type           100000 non-null  int64
 12  fare_amount            100000 non-null  float64
 13  extra                  100000 non-null  float64
 14  mta_tax                100000 non-null  float64
 15  tip_amount             100000 non-null  float64
 16  tolls_amount           100000 non-null  float64
 17  improvement_surcharge  100000 non-null  float64
 18  total_amount           100000 non-null  float64
 19  trip_id                100000 non-null  int64
dtypes: datetime64[ns](2), float64(12), int64(5), object(1)
memory usage: 15.3+ MB
```

```
# checking duplicates
ans = int(data.duplicated().sum())
ans
```

```
0
```

## Data Transformation and Modeling Done Together

```
# datetime_dim
datetime_dim = data[['tpep_pickup_datetime', 'tpep_dropoff_datetime']]
```

```
datetime_dim['pickup_hour'] = datetime_dim['tpep_pickup_datetime'].dt.hour
datetime_dim['pickup_day'] = datetime_dim['tpep_pickup_datetime'].dt.day
datetime_dim['pickup_weekday'] = datetime_dim['tpep_pickup_datetime'].dt.weekday
datetime_dim['pickup_month'] = datetime_dim['tpep_pickup_datetime'].dt.month
datetime_dim['pickup_year'] = datetime_dim['tpep_pickup_datetime'].dt.year
```

```
/tmp/ipython-input-1816695665.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  datetime_dim['pickup_hour'] = datetime_dim['tpep_pickup_datetime'].dt.hour
/tmp/ipython-input-1816695665.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  datetime_dim['pickup_day'] = datetime_dim['tpep_pickup_datetime'].dt.day
/tmp/ipython-input-1816695665.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  datetime_dim['pickup_weekday'] = datetime_dim['tpep_pickup_datetime'].dt.weekday
/tmp/ipython-input-1816695665.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  datetime_dim['pickup_month'] = datetime_dim['tpep_pickup_datetime'].dt.month
/tmp/ipython-input-1816695665.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  datetime_dim['pickup_year'] = datetime_dim['tpep_pickup_datetime'].dt.year
```

```
datetime_dim['drop_hour'] = datetime_dim['tpep_dropoff_datetime'].dt.hour
datetime_dim['drop_day'] = datetime_dim['tpep_dropoff_datetime'].dt.day
datetime_dim['drop_weekday'] = datetime_dim['tpep_dropoff_datetime'].dt.weekday
datetime_dim['drop_month'] = datetime_dim['tpep_dropoff_datetime'].dt.month
datetime_dim['drop_year'] = datetime_dim['tpep_dropoff_datetime'].dt.year
```

```
/tmp/ipython-input-2369486348.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  datetime_dim['drop_hour'] = datetime_dim['tpep_dropoff_datetime'].dt.hour
/tmp/ipython-input-2369486348.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  datetime_dim['drop_day'] = datetime_dim['tpep_dropoff_datetime'].dt.day
/tmp/ipython-input-2369486348.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  datetime_dim['drop_weekday'] = datetime_dim['tpep_dropoff_datetime'].dt.weekday
/tmp/ipython-input-2369486348.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  datetime_dim['drop_month'] = datetime_dim['tpep_dropoff_datetime'].dt.month
/tmp/ipython-input-2369486348.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  datetime_dim['drop_year'] = datetime_dim['tpep_dropoff_datetime'].dt.year
```

```
datetime_dim['datetime_id'] = datetime_dim.index
```

```
/tmp/ipython-input-1085316748.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  datetime_dim['datetime_id'] = datetime_dim.index
```

```
datetime_dim.columns
```

```
Index(['tpep_pickup_datetime', 'tpep_dropoff_datetime', 'pickup_hour',
       'pickup_day', 'pickup_weekday', 'pickup_month', 'pickup_year',
       'drop_hour', 'drop_day', 'drop_weekday', 'drop_month', 'drop_year',
       'datetime_id'],
      dtype='object')
```

Start coding or generate with AI.

```
pickup_location_dim = data[['pickup_latitude', 'pickup_longitude']]
pickup_location_dim['pickup_location_id'] = pickup_location_dim.index
pickup_location_dim = pickup_location_dim[['pickup_location_id','pickup_latitude','pickup_longitude']]
pickup_location_dim
```

```
/tmp/ipython-input-3590796290.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  pickup_location_dim['pickup_location_id'] = pickup_location_dim.index
```

| | pickup_location_id | pickup_latitude | pickup_longitude |
|---|---|---|---|
| **0** | 0 | 40.765152 | -73.976746 |
| **1** | 1 | 40.767925 | -73.983482 |
| **2** | 2 | 40.644810 | -73.782021 |
| **3** | 3 | 40.769814 | -73.863419 |
| **4** | 4 | 40.792183 | -73.971741 |
| **...** | ... | ... | ... |
| **99995** | 99995 | 40.750519 | -73.990898 |
| **99996** | 99996 | 40.718296 | -74.014488 |
| **99997** | 99997 | 40.774097 | -73.963379 |
| **99998** | 99998 | 40.763111 | -73.984901 |
| **99999** | 99999 | 40.750473 | -73.990685 |

100000 rows × 3 columns

Next steps: Generate code with `pickup_location_dim` View recommended plots New interactive sheet

```
dropoff_location_dim = data[['dropoff_longitude', 'dropoff_latitude']]
dropoff_location_dim['dropoff_location_id'] = dropoff_location_dim.index
dropoff_location_dim = dropoff_location_dim[['dropoff_location_id', 'dropoff_latitude', 'dropoff_longitude']]
dropoff_location_dim
```

```
/tmp/ipython-input-340585180.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  dropoff_location_dim['dropoff_location_id'] = dropoff_location_dim.index
```

| | dropoff_location_id | dropoff_latitude | dropoff_longitude |
|---|---|---|---|
| **0** | 0 | 40.746128 | -74.004265 |
| **1** | 1 | 40.733166 | -74.005943 |
| **2** | 2 | 40.675770 | -73.974541 |
| **3** | 3 | 40.757767 | -73.969650 |
| **4** | 4 | 40.695053 | -74.177170 |
| **...** | ... | ... | ... |
| **99995** | 99995 | 40.750462 | -73.998245 |
| **99996** | 99996 | 40.752529 | -73.982361 |
| **99997** | 99997 | 40.770512 | -73.865028 |
| **99998** | 99998 | 40.759148 | -73.970695 |
| **99999** | 99999 | 40.754910 | -73.980354 |

100000 rows × 3 columns

Next steps: Generate code with `dropoff_location_dim` View recommended plots New interactive sheet

```
payment_type_name = {
    1:"Credit card",
    2:"Cash",
    3:"No charge",
    4:"Dispute",
    5:"Unknown",
    6:"Voided trip"
}

payment_type_dim = data[['payment_type']]
payment_type_dim['payment_type_id'] = payment_type_dim.index
payment_type_dim['payment_type_name'] = payment_type_dim['payment_type'].map(payment_type_name)
payment_type_dim
```

```
/tmp/ipython-input-3171444021.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  payment_type_dim['payment_type_id'] = payment_type_dim.index
/tmp/ipython-input-3171444021.py:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  payment_type_dim['payment_type_name'] = payment_type_dim['payment_type'].map(payment_type_name)
```

| | payment_type | payment_type_id | payment_type_name |
|---|---|---|---|
| **0** | 1 | 0 | Credit card |
| **1** | 1 | 1 | Credit card |
| **2** | 1 | 2 | Credit card |
| **3** | 1 | 3 | Credit card |
| **4** | 1 | 4 | Credit card |
| **...** | ... | ... | ... |
| **99995** | 2 | 99995 | Cash |
| **99996** | 1 | 99996 | Credit card |
| **99997** | 1 | 99997 | Credit card |
| **99998** | 1 | 99998 | Credit card |
| **99999** | 2 | 99999 | Cash |

100000 rows × 3 columns

Next steps: Generate code with payment_type_dim View recommended plots New interactive sheet

```
rate_code_type = {
    1:"Standard rate",
    2:"JFK",
    3:"Newark",
    4:"Nassau or Westchester",
    5:"Negotiated fare",
    6:"Group ride"
}

rate_code_dim = data[['RatecodeID']]
rate_code_dim['rate_code_id'] = rate_code_dim.index
rate_code_dim['rate_code_type'] = rate_code_dim['RatecodeID'].map(rate_code_type)
rate_code_dim = rate_code_dim[['rate_code_id', 'RatecodeID', 'rate_code_type']]
rate_code_dim
```

```
/tmp/ipython-input-2466067013.py:11: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  rate_code_dim['rate_code_id'] = rate_code_dim.index
/tmp/ipython-input-2466067013.py:12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  rate_code_dim['rate_code_type'] = rate_code_dim['RatecodeID'].map(rate_code_type)
```

| | rate_code_id | RatecodeID | rate_code_type |
|---|---|---|---|
| **0** | 0 | 1 | Standard rate |
| **1** | 1 | 1 | Standard rate |
| **2** | 2 | 1 | Standard rate |
| **3** | 3 | 1 | Standard rate |
| **4** | 4 | 3 | Newark |
| **...** | ... | ... | ... |
| **99995** | 99995 | 1 | Standard rate |
| **99996** | 99996 | 1 | Standard rate |
| **99997** | 99997 | 1 | Standard rate |
| **99998** | 99998 | 1 | Standard rate |
| **99999** | 99999 | 1 | Standard rate |

100000 rows × 3 columns

Next steps: Generate code with rate_code_dim View recommended plots New interactive sheet

```
passenger_count_dim = data[['passenger_count']]
passenger_count_dim['passenger_count_id'] = passenger_count_dim.index
passenger_count_dim = passenger_count_dim[['passenger_count_id','passenger_count']]

trip_distance_dim = data[['trip_distance']].reset_index(drop=True)
trip_distance_dim['trip_distance_id'] = trip_distance_dim.index
trip_distance_dim = trip_distance_dim[['trip_distance_id','trip_distance']]
```

```
/tmp/ipython-input-764414733.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  passenger_count_dim['passenger_count_id'] = passenger_count_dim.index
```

```
Start coding or generate with AI.
```

```
# merging columns by using trip_id everytime with other mentioned indexed id of extrcated data

fact_table = data.merge(passenger_count_dim, left_on='trip_id', right_on='passenger_count_id') \
             .merge(trip_distance_dim, left_on='trip_id', right_on='trip_distance_id') \
             .merge(rate_code_dim, left_on='trip_id', right_on='rate_code_id') \
             .merge(pickup_location_dim, left_on='trip_id', right_on='pickup_location_id') \
             .merge(dropoff_location_dim, left_on='trip_id', right_on='dropoff_location_id')\
             .merge(datetime_dim, left_on='trip_id', right_on='datetime_id') \
             .merge(payment_type_dim, left_on='trip_id', right_on='payment_type_id') \
             [['trip_id','VendorID', 'datetime_id', 'passenger_count_id',
               'trip_distance_id', 'rate_code_id', 'store_and_fwd_flag', 'pickup_location_id', 'dropoff_location_id',
               'payment_type_id', 'fare_amount', 'extra', 'mta_tax', 'tip_amount', 'tolls_amount',
               'improvement_surcharge', 'total_amount']]
```

```
fact_table
```

| | trip_id | VendorID | datetime_id | passenger_count_id | trip_distance_id | rate_code_id | store_and_fwd_flag | pickup_location_id | d |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 0 | 0 | 0 | 0 | N | 0 | |
| **1** | 1 | 1 | 1 | 1 | 1 | 1 | N | 1 | |
| **2** | 2 | 2 | 2 | 2 | 2 | 2 | N | 2 | |
| **3** | 3 | 2 | 3 | 3 | 3 | 3 | N | 3 | |
| **4** | 4 | 2 | 4 | 4 | 4 | 4 | N | 4 | |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **99995** | 99995 | 1 | 99995 | 99995 | 99995 | 99995 | N | 99995 | |
| **99996** | 99996 | 1 | 99996 | 99996 | 99996 | 99996 | N | 99996 | |
| **99997** | 99997 | 1 | 99997 | 99997 | 99997 | 99997 | N | 99997 | |
| **99998** | 99998 | 2 | 99998 | 99998 | 99998 | 99998 | N | 99998 | |
| **99999** | 99999 | 1 | 99999 | 99999 | 99999 | 99999 | N | 99999 | |

100000 rows × 17 columns

Next steps: Generate code with fact_table View recommended plots New interactive sheet

```
# List all variables in memory
%who DataFrame
```

```
data     datetime_dim     dropoff_location_dim     fact_table     passenger_count_dim     payment_type_dim     pickup_location_di
```

```
Start coding or generate with AI.
```

## Data Loading

```
# pip install duckdb
```

```
import duckdb
```

```
con = duckdb.connect("uber_etl.duckdb")  # or use ":memory:" for in-memory
```

```
con.register("fact_df", fact_table)  # register DataFrame as a DuckDB table
con.execute("CREATE TABLE fact_table AS SELECT * FROM fact_df")
```

```
<duckdb.duckdb.DuckDBPyConnection at 0x7e9cd5b31870>
```

```
# checking the result
result = con.execute("SELECT * FROM fact_table LIMIT 5").fetchdf()
result
```

| | trip_id | VendorID | datetime_id | passenger_count_id | trip_distance_id | rate_code_id | store_and_fwd_flag | pickup_location_id | dropo |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 1 | 0 | 0 | 0 | 0 | N | 0 | |
| **1** | 1 | 1 | 1 | 1 | 1 | 1 | N | 1 | |
| **2** | 2 | 2 | 2 | 2 | 2 | 2 | N | 2 | |
| **3** | 3 | 2 | 3 | 3 | 3 | 3 | N | 3 | |
| **4** | 4 | 2 | 4 | 4 | 4 | 4 | N | 4 | |

Next steps: Generate code with result View recommended plots New interactive sheet

```
# our dataFrames
dfs = {
    "passenger_count": passenger_count_dim,
    "trip_distance": trip_distance_dim,
    "rate_code": rate_code_dim,
    "pickup_location": pickup_location_dim,
    "dropoff_location": dropoff_location_dim,
    "datetime": datetime_dim,
    "payment_type": payment_type_dim
}

# Register and create tables in DuckDB
for table_name, df in dfs.items():
    con.register(f"{table_name}_df", df)  # Temporary name in DuckDB
    con.execute(f"CREATE TABLE {table_name} AS SELECT * FROM {table_name}_df")
```

```
Start coding or generate with AI.
```

## Data Analysis using SQL Queries

```
result = con.execute('Select * from payment_type_df limit 4').fetch_df()
result
```

| | payment_type | payment_type_id | payment_type_name |
|---|---|---|---|
| **0** | 1 | 0 | Credit card |
| **1** | 1 | 1 | Credit card |
| **2** | 1 | 2 | Credit card |
| **3** | 1 | 3 | Credit card |

Next steps: Generate code with result View recommended plots New interactive sheet

```
# trips by passengers count
con.execute("""
    SELECT pc.passenger_count, COUNT(*) AS trip_count
    FROM fact_table ft
    JOIN passenger_count pc ON ft.passenger_count_id = pc.passenger_count_id
    GROUP BY pc.passenger_count
    ORDER BY trip_count DESC
""").fetchdf()
```

| | passenger_count | trip_count |
|---|---|---|
| **0** | 1 | 65493 |
| **1** | 2 | 13709 |
| **2** | 5 | 8748 |
| **3** | 6 | 6077 |
| **4** | 3 | 4076 |
| **5** | 4 | 1894 |
| **6** | 0 | 3 |

```
# top-5 highest fare trips
con.execute("""
    SELECT * FROM fact_table
    ORDER BY fare_amount DESC
    LIMIT 5
""").fetchdf()
```

| | trip_id | VendorID | datetime_id | passenger_count_id | trip_distance_id | rate_code_id | store_and_fwd_flag | pickup_location_id | dropo |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 75653 | 1 | 75653 | 75653 | 75653 | 75653 | N | 75653 | |
| **1** | 76891 | 2 | 76891 | 76891 | 76891 | 76891 | N | 76891 | |
| **2** | 77168 | 2 | 77168 | 77168 | 77168 | 77168 | N | 77168 | |
| **3** | 35136 | 2 | 35136 | 35136 | 35136 | 35136 | N | 35136 | |
| **4** | 82857 | 1 | 82857 | 82857 | 82857 | 82857 | N | 82857 | |

```
# average trip distance by rate code
con.execute("""
    SELECT rc.rate_code_type, AVG(td.trip_distance) AS avg_distance
    FROM fact_table ft
    JOIN rate_code rc ON ft.rate_code_id = rc.rate_code_id
    JOIN trip_distance td ON ft.trip_distance_id = td.trip_distance_id
    GROUP BY rc.rate_code_type
""").fetchdf()
```

```
              rate_code_type  avg_distance
0                     Newark     16.432290
1      Nassau or Westchester     20.199792
2                 Group ride      0.100000
3              Standard rate      2.653575
4                        JFK     17.446343
5             Negotiated fare     6.089081
```

```
# trips per day
con.execute("""
    SELECT d.pickup_day, COUNT(*) AS trips
    FROM fact_table ft
    JOIN datetime d ON ft.datetime_id = d.datetime_id
    GROUP BY d.pickup_day
    ORDER BY d.pickup_day
""").fetchdf()
```

```
   pickup_day  trips
0           1  23220
1          10  76780
```

```
# fetching all payment_types ordered by payment_type
con.execute("""
  select * from payment_type_df p order by payment_type;
""").fetch_df()
```

```
       payment_type  payment_type_id  payment_type_name
0                 1                0        Credit card
1                 1                1        Credit card
2                 1                2        Credit card
3                 1                3        Credit card
4                 1                4        Credit card
...             ...              ...                ...
99995             4            92629            Dispute
99996             4            92805            Dispute
99997             4            93586            Dispute
99998             4            95234            Dispute
99999             4            95764            Dispute
100000 rows × 3 columns
```

Start coding or generate with AI.

## Some Aggregations using SQL

```
# total revenue by payment type
con.execute("""
    SELECT pt.payment_type_name, SUM(ft.total_amount) AS total_revenue
    FROM fact_table ft
    JOIN payment_type pt ON ft.payment_type_id = pt.payment_type_id
    GROUP BY pt.payment_type_name
    ORDER BY total_revenue DESC
""").fetchdf()
```

| | payment_type_name | total_revenue |
|---|---|---|
| 0 | Credit card | 1202467.81 |
| 1 | Cash | 434002.97 |
| 2 | No charge | 1838.95 |
| 3 | Dispute | 762.36 |

```
# average tip per passenger count
con.execute("""
    SELECT pc.passenger_count, ROUND(AVG(ft.tip_amount), 2) AS avg_tip
    FROM fact_table ft
    JOIN passenger_count pc ON ft.passenger_count_id = pc.passenger_count_id
    GROUP BY pc.passenger_count
    ORDER BY pc.passenger_count
""").fetchdf()
```

| | passenger_count | avg_tip |
|---|---|---|
| 0 | 0 | 0.67 |
| 1 | 1 | 1.88 |
| 2 | 2 | 1.88 |
| 3 | 3 | 1.84 |
| 4 | 4 | 1.72 |
| 5 | 5 | 1.92 |
| 6 | 6 | 1.81 |

```
# total bills paid by rate code
con.execute("""
    SELECT rc.rate_code_type, SUM(ft.tolls_amount) AS total_tolls
    FROM fact_table ft
    JOIN rate_code rc ON ft.rate_code_id = rc.rate_code_id
    GROUP BY rc.rate_code_type
    ORDER BY total_tolls DESC
""").fetchdf()
```

| | rate_code_type | total_tolls |
|---|---|---|
| 0 | Standard rate | 22577.31 |
| 1 | JFK | 9815.75 |
| 2 | Newark | 3380.18 |
| 3 | Negotiated fare | 908.01 |
| 4 | Nassau or Westchester | 60.20 |
| 5 | Group ride | 0.00 |

```
# average fare by day of week
con.execute("""
    SELECT d.pickup_weekday, ROUND(AVG(ft.fare_amount), 2) AS avg_fare
    FROM fact_table ft
    JOIN datetime_df d ON ft.datetime_id = d.datetime_id
    GROUP BY d.pickup_weekday
    ORDER BY d.pickup_weekday
""").fetch_df()
```

| | pickup_weekday | avg_fare |
|---|---|---|
| 0 | 1 | 14.08 |
| 1 | 3 | 13.00 |

```
datetime_dim.columns
```

```
Index(['tpep_pickup_datetime', 'tpep_dropoff_datetime', 'pickup_hour',
       'pickup_day', 'pickup_weekday', 'pickup_month', 'pickup_year',
       'drop_hour', 'drop_day', 'drop_weekday', 'drop_month', 'drop_year',
       'datetime_id'],
      dtype='object')
```

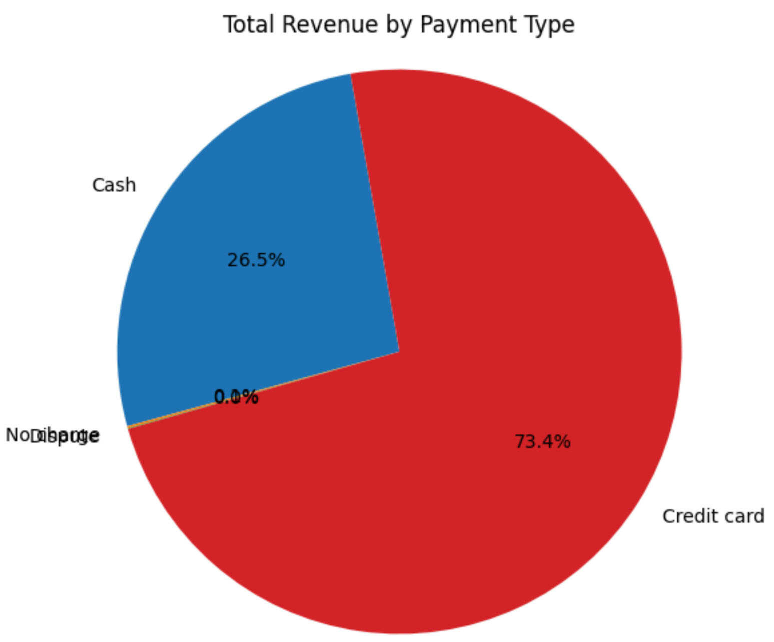
```
Start coding or generate with AI.
```

## Data Visualizations

```
# Total Revenue by Payment Type (Pie Chart)
import matplotlib.pyplot as plt

df_payment = con.execute("""
    SELECT pt.payment_type_name, SUM(ft.total_amount) AS total_revenue
    FROM fact_table ft
    JOIN payment_type pt ON ft.payment_type_id = pt.payment_type_id
    GROUP BY pt.payment_type_name
""").fetchdf()

# Plot
plt.figure(figsize=(6, 6))
plt.pie(df_payment['total_revenue'], labels=df_payment['payment_type_name'], autopct='%1.1f%%', startangle=100)
plt.title("Total Revenue by Payment Type")
plt.axis('equal')
plt.show()
```



```
# Average Fare by Day of Week (Line Plot)
df_day_fare = con.execute("""
    SELECT d.pickup_weekday, ROUND(AVG(ft.fare_amount), 2) AS avg_fare
    FROM fact_table ft
    JOIN datetime d ON ft.datetime_id = d.datetime_id
    GROUP BY d.pickup_weekday
    ORDER BY d.pickup_weekday
""").fetchdf()

# Plot
plt.figure(figsize=(8, 5))
plt.plot(df_day_fare['pickup_weekday'], df_day_fare['avg_fare'], marker='o', color='green')
plt.title("Average Fare by Day of Week")
plt.xlabel("Day of Week")
plt.ylabel("Average Fare ($)")
plt.grid(True, linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```

Average Fare by Day of Week