datascience_assignment_14.1

July 9, 2018

0.1 1. Create an sqlalchemy engine using a sample from the data set

0.2 Steps:

- Import all the packages needed from sqlalchemy
- Create Base and Engine
- Creae a class Adult with table name 'Adult' and fields: 'age', 'workclass', 'fnlwgt', 'education', 'education_num', 'marital_status', 'occupation', 'relationship', 'race', 'sex', 'capital_gain', 'capital_loss', 'hours_per_week', 'native_country', 'income'
- Create a session and bind to Engine
- Insert five records from the sample given and commit the session

```
In [1]: #Import all the packages needed from sqlalchemy
        from sqlalchemy import create_engine
        from sqlalchemy.ext.declarative import declarative_base
        from sqlalchemy import Column, Integer, String
        from sqlalchemy.orm import sessionmaker
        from sqlalchemy.ext.declarative import declarative_base
        from sqlalchemy import func
```

```
In [2]: # Create Base and Engine
        Base = declarative_base()
        engine = create_engine('sqlite:///:memory:', echo=True)
```

```
In [3]: class Adult(Base):
        ...     __tablename__ = 'Adult'
        ...
        ...     id = Column(Integer, primary_key=True)
        ...     age = Column(Integer)
        ...     workclass = Column(String)
        ...     fnlwgt = Column(String)
        ...     education = Column(String)
        ...     education_num = Column(String)
        ...     marital_status = Column(String)
        ...     occupation = Column(String)
        ...     relationship = Column(String)
        ...     race = Column(String)
        ...     sex = Column(String)
        ...     capital_gain = Column(Integer)
        ...     capital_loss = Column(Integer)
        ...     hours_per_week = Column(Integer)
        ...     native_country = Column(String)
        ...     income = Column(String)
        ...
        ...     def __repr__(self):
        ...         return "<Adult(age='%d', workclass='%s', fnlwgt='%s, education='%s', educa
        ...         self.age, self.workclass, self.fnlwgt, self.education, self.education_num,
```

```
In [4]: # Dipslay the Columns of Adult table
        Adult.__table__
```

```
Out[4]: Table('Adult', MetaData(bind=None), Column('id', Integer(), table=<Adult>, primary_key
```

```
In [5]: # Create Adult table
        Base.metadata.create_all(engine)
```

```
2018-07-08 22:53:35,316 INFO sqlalchemy.engine.base.Engine SELECT CAST('test plain returns' AS
2018-07-08 22:53:35,322 INFO sqlalchemy.engine.base.Engine ()
2018-07-08 22:53:35,324 INFO sqlalchemy.engine.base.Engine SELECT CAST('test unicode returns'
2018-07-08 22:53:35,325 INFO sqlalchemy.engine.base.Engine ()
2018-07-08 22:53:35,327 INFO sqlalchemy.engine.base.Engine PRAGMA table_info("Adult")
2018-07-08 22:53:35,327 INFO sqlalchemy.engine.base.Engine ()
2018-07-08 22:53:35,329 INFO sqlalchemy.engine.base.Engine
CREATE TABLE "Adult" (
        id INTEGER NOT NULL,
        age INTEGER,
        workclass VARCHAR,
        fnlwgt VARCHAR,
        education VARCHAR,
        education_num VARCHAR,
        marital_status VARCHAR,
        occupation VARCHAR,
        relationship VARCHAR,
        race VARCHAR,
        sex VARCHAR,
        capital_gain INTEGER,
        capital_loss INTEGER,
        hours_per_week INTEGER,
        native_country VARCHAR,
        income VARCHAR,
        PRIMARY KEY (id)
)


2018-07-08 22:53:35,330 INFO sqlalchemy.engine.base.Engine ()
2018-07-08 22:53:35,331 INFO sqlalchemy.engine.base.Engine COMMIT
```

```
In [6]: # Create a Session and bind to engine
        Session = sessionmaker(bind=engine)
        session = Session()

        # Add five records from the sample provided
        session.add_all([Adult(age='39', workclass='State-gov', fnlwgt='77516', education='Bac
                         Adult(age=50, workclass='Self-emp-not-inc', fnlwgt='83311', education
                         Adult(age=53, workclass='Private', fnlwgt='234721', education='11th',
                         Adult(age=52, workclass='Self-emp-not-inc', fnlwgt='209642', educatio
                         Adult(age=23, workclass='Private', fnlwgt='122272', education='Bachel
                        ])

        # Commit the inserted records
        session.commit()
```

```
2018-07-08 22:53:38,747 INFO sqlalchemy.engine.base.Engine BEGIN (implicit)
2018-07-08 22:53:38,750 INFO sqlalchemy.engine.base.Engine INSERT INTO "Adult" (age, workclass
2018-07-08 22:53:38,751 INFO sqlalchemy.engine.base.Engine ('39', 'State-gov', '77516', 'Bache
2018-07-08 22:53:38,752 INFO sqlalchemy.engine.base.Engine INSERT INTO "Adult" (age, workclass
2018-07-08 22:53:38,753 INFO sqlalchemy.engine.base.Engine (50, 'Self-emp-not-inc', '83311', '
2018-07-08 22:53:38,754 INFO sqlalchemy.engine.base.Engine INSERT INTO "Adult" (age, workclass
2018-07-08 22:53:38,755 INFO sqlalchemy.engine.base.Engine (53, 'Private', '234721', '11th', '
2018-07-08 22:53:38,756 INFO sqlalchemy.engine.base.Engine INSERT INTO "Adult" (age, workclass
2018-07-08 22:53:38,757 INFO sqlalchemy.engine.base.Engine (52, 'Self-emp-not-inc', '209642',
2018-07-08 22:53:38,758 INFO sqlalchemy.engine.base.Engine INSERT INTO "Adult" (age, workclass
2018-07-08 22:53:38,758 INFO sqlalchemy.engine.base.Engine (23, 'Private', '122272', 'Bachelor
2018-07-08 22:53:38,759 INFO sqlalchemy.engine.base.Engine COMMIT
```

0.3 4. Write two filter queries

0.4 Steps:

- Write a filter query for Adult whose income is '>50K' and assign to person_with_income_greater_50k
- Write a filter query for Adult whose age is less than 25 and assign to person_with_age_less_than_25

```
In [7]: # Filter query for adult whose income is '>50K'
        person_with_income_greater_50k = session.query(Adult).filter_by(income='>50K').one()
        print(person_with_income_greater_50k)
```

```
2018-07-08 22:53:52,687 INFO sqlalchemy.engine.base.Engine BEGIN (implicit)
2018-07-08 22:53:52,690 INFO sqlalchemy.engine.base.Engine SELECT "Adult".id AS "Adult_id", "A
FROM "Adult"
WHERE "Adult".income = ?
2018-07-08 22:53:52,691 INFO sqlalchemy.engine.base.Engine ('>50K',)
<Adult(age='52', workclass='Self-emp-not-inc', fnlwgt='209642', education='HS-grad', education_
In [8]: # Filter query for adult whose age is less than 25
        from sqlalchemy import text
        person_with_age_less_than_25 = session.query(Adult).filter(text("age<25")).one()
        print(person_with_age_less_than_25)

2018-07-08 22:54:00,096 INFO sqlalchemy.engine.base.Engine SELECT "Adult".id AS "Adult_id", "Ad
FROM "Adult"
WHERE age<25
2018-07-08 22:54:00,097 INFO sqlalchemy.engine.base.Engine ()
<Adult(age='23', workclass='Private', fnlwgt='122272, education='Bachelors', education_num='13
```

0.5 5. Write two function queries

0.6 Steps:

- Write a function query to return count of adults from Adult table group by marital_status
- Write a function query to total number of records in Adult table

```
In [9]: # Write a function query to return count of adults group by martial_status
        session.query(Adult.marital_status, func.count(Adult.marital_status)).group_by(Adult.ma

2018-07-08 22:54:07,059 INFO sqlalchemy.engine.base.Engine SELECT "Adult".marital_status AS "Ad
FROM "Adult" GROUP BY "Adult".marital_status
2018-07-08 22:54:07,060 INFO sqlalchemy.engine.base.Engine ()


Out[9]: [('Husband', 2), ('Not-in-family', 2), ('Own-child', 1)]

In [10]: #Write a function query to total number of records in Adult table
         total_records = session.query(func.count(Adult.id)).scalar()
         print("Total number of records = " + str(total_records))

2018-07-08 22:54:10,763 INFO sqlalchemy.engine.base.Engine SELECT count("Adult".id) AS count_1
FROM "Adult"
2018-07-08 22:54:10,764 INFO sqlalchemy.engine.base.Engine ()
Total number of records = 5
```

0.7 2. Write two basic update queries

0.8 Steps:

- Update person_with_income_greater_50k by changing capital_loss field to 500, hours_per_week field to 42
- Update person_with_age_less_than_25 by changing occupation to 'Exec-managerial'
- Commit the session for the updated records
- Query again for both person_with_income_greater_50k and person_with_age_less_than_25 and check that update has happened

```
In [11]: # Update person_with_income_greater_50k by changing capital_loss field to 500,
         # hours_per_week field to 42
         person_with_income_greater_50k.capital_loss=500
         person_with_income_greater_50k.hours_per_week=42

         # Update person_with_age_less_than_25 by changing occupation to 'Exec-managerial'
         person_with_age_less_than_25.occupation='Exec-managerial'

         # Commit the session for the updated records
         session.commit()
```

```
2018-07-08 22:54:26,878 INFO sqlalchemy.engine.base.Engine UPDATE "Adult" SET capital_loss=?, h
2018-07-08 22:54:26,880 INFO sqlalchemy.engine.base.Engine (500, 42, 4)
2018-07-08 22:54:26,881 INFO sqlalchemy.engine.base.Engine UPDATE "Adult" SET occupation=? WHEF
2018-07-08 22:54:26,882 INFO sqlalchemy.engine.base.Engine ('Exec-managerial', 5)
2018-07-08 22:54:26,883 INFO sqlalchemy.engine.base.Engine COMMIT
```

```
In [12]: # Query again for both person_with_income_greater_50k and check that capital_loss and
         # are updated
         person_with_income_greater_50k = session.query(Adult).filter_by(income='>50K').one()
         print(person_with_income_greater_50k)
```

```
2018-07-08 22:54:30,015 INFO sqlalchemy.engine.base.Engine BEGIN (implicit)
2018-07-08 22:54:30,017 INFO sqlalchemy.engine.base.Engine SELECT "Adult".id AS "Adult_id", "Ad
FROM "Adult"
WHERE "Adult".income = ?
2018-07-08 22:54:30,018 INFO sqlalchemy.engine.base.Engine ('>50K',)
<Adult(age='52', workclass='Self-emp-not-inc', fnlwgt='209642, education='HS-grad', education_n
```

```
In [13]: # Query again for both erson_with_age_less_than_25 and check that occupation field is
         person_with_age_less_than_25 = session.query(Adult).filter(text("age<25")).one()
         print(person_with_age_less_than_25)
```

```
2018-07-08 22:54:33,620 INFO sqlalchemy.engine.base.Engine SELECT "Adult".id AS "Adult_id", "Ad
FROM "Adult"
WHERE age<25
2018-07-08 22:54:33,621 INFO sqlalchemy.engine.base.Engine ()
<Adult(age='23', workclass='Private', fnlwgt='122272, education='Bachelors', education_num='13
```

0.9 3. Write two delete queries

0.10 Steps:

- Delete person_with_income_greater_50k and person_with_age_less_than_25
- Commit the session
- Check that deleted records are no longer available

```
In [14]: # Delete records person_with_income_greater_50k and person_with_age_less_than_25
         session.delete(person_with_income_greater_50k)
         session.delete(person_with_age_less_than_25)

         # Commit the session
         session.commit()
```

```
2018-07-08 22:54:39,389 INFO sqlalchemy.engine.base.Engine DELETE FROM "Adult" WHERE "Adult".i
2018-07-08 22:54:39,390 INFO sqlalchemy.engine.base.Engine ((4,), (5,))
2018-07-08 22:54:39,392 INFO sqlalchemy.engine.base.Engine COMMIT
```

```
In [17]: # Query the DB again and check that there are no record for person_with_income_greate
         # It will throw an Exception NoResultFound
         person_with_income_greater_50k = session.query(Adult).filter_by(income='>50K').one()
         print(person_with_income_greater_50k)
```

```
2018-07-08 22:55:33,937 INFO sqlalchemy.engine.base.Engine SELECT "Adult".id AS "Adult_id", "A
FROM "Adult"
WHERE "Adult".income = ?
2018-07-08 22:55:33,939 INFO sqlalchemy.engine.base.Engine ('>50K',)
```

```
        ---------------------------------------------------------------------------

        NoResultFound                             Traceback (most recent call last)

        <ipython-input-17-b4a4c46e964b> in <module>()
          1 # Query the DB again and check that there are no record for person_with_income_gre
          2 # It will throw an Exception NoResultFound
    ----> 3 person_with_income_greater_50k = session.query(Adult).filter_by(income='>50K').one
          4 print(person_with_income_greater_50k)


        E:\anaconda\lib\site-packages\sqlalchemy\orm\query.py in one(self)
       2841         else:
       2842             if ret is None:
    -> 2843                 raise orm_exc.NoResultFound("No row was found for one()")
       2844             return ret
       2845


        NoResultFound: No row was found for one()
```

```
In [18]: # Query the DB again and check that there are no record for person_with_age_less_than
         # # It will throw an Exception NoResultFound
         person_with_age_less_than_25 = session.query(Adult).filter(text("age<25")).one()
         print(person_with_age_less_than_25)
```

```
2018-07-08 22:55:48,858 INFO sqlalchemy.engine.base.Engine SELECT "Adult".id AS "Adult_id", "Ad
FROM "Adult"
WHERE age<25
2018-07-08 22:55:48,859 INFO sqlalchemy.engine.base.Engine ()
```

```
        ---------------------------------------------------------------------------

        NoResultFound                             Traceback (most recent call last)

        <ipython-input-18-770536fce7fc> in <module>()
          1 # Query the DB again and check that there are no record for person_with_age_less_th
          2 # # It will throw an Exception NoResultFound
    ----> 3 person_with_age_less_than_25 = session.query(Adult).filter(text("age<25")).one()
          4 print(person_with_age_less_than_25)


        E:\anaconda\lib\site-packages\sqlalchemy\orm\query.py in one(self)
       2841         else:
       2842             if ret is None:
    -> 2843                 raise orm_exc.NoResultFound("No row was found for one()")
       2844             return ret
       2845


        NoResultFound: No row was found for one()
```