Database e Data Analytics - Laboratorio

ESERCITAZIONE 2 - Soluzione

```
url = f"postgresql+psycopg2://{username}:{password}@{host}:{port}/{nomeDb}"
engine = create_engine(url, echo=True)
metadata = MetaData(schema='segreteria', bind=engine)
studenti = Table('studenti', metadata, autoload=True)
esami = Table('esami', metadata, autoload=True
corsi = Table('corsi', metadata, autoload=True
docenti = Table('docenti', metadata, autoload=True)
with engine.connect() as connection:
    query = * query sql alchemy*
    content = connection.execute(query)
for item in content: print(item)
```

Prima parte

1.a) *dati relativi agli esami il cui voto è minore di 21 o maggiore di 27*

SQL:

```
select * from segreteria.esami
    where voto < 21 or voto > 27;
```

SQLAlchemy:

```
select(esami).\
    where((esami.c.voto < 21) | (esami.c.voto > 27))

select(esami).\
    where(or_(esami.c.voto < 21, esami.c.voto > 27))
```

1.b) *codice e cognome dei docenti di cui non è noto il numero di telefono*

SQL:

```
select cod_docente, cognome from segreteria.docenti
    where num_telefono is null;
```

SQLAlchemy:

```
select(docenti.c.cod_docente, docenti.c.cognome).\
    where(docenti.c.num_telefono == None)
```

1.c) *matricola degli studenti il cui cognome inizia con 'M' o 'N' e termina con 'i'*

SQL:

```
select matricola from segreteria.studenti
    where cognome like 'M%i' or cognome like 'N%i';
```

ASQLAlchemy:

```
select(studenti.c.matricola).\
    where((studenti.c.cognome.like('M%i')) | (studenti.c.cognome.like('N%i')))
```

```
select(studenti.c.matricola).\
        where(or_(studenti.c.cognome.like('M%i'),studenti.c.cognome.like('N%i')))
```

1.d) *matricola degli studenti che hanno sostenuto nel 2006 o nel 2007 un esame con voto pari a 30 o 33 (lode)*

SQL:

```
select studente from segreteria.esami
    where data between '2006-01-01' and '2007-12-31' and voto in (30, 33);

select studente from segreteria.esami
    where data between '2006-01-01' and '2007-12-31' and (voto = 30 or voto = 33);
```

SQLAlchemy:

```
select(esami.c.studente).\
   where((esami.c.data.between('2006-01-01','2007-12-31'))&\
        ((esami.c.voto == 30) | (esami.c.voto == 33)))
```

1.e) *cognome e nome degli studenti nati prima del 1984, con l'eliminazione di eventuali duplicati*

SQL:

```
select distinct cognome, nome from segreteria.studenti
    where data_nascita < '1984-01-01';
```

SQLAlchemy:

```
select(studenti.c.cognome,studenti.c.nome).\
        where(studenti.c.data_nascita < '1984-01-01').distinct()
```

1.f) *dati di tutti gli studenti del biennio, ordinati in modo decrescente rispetto all'età e, a parità di età, ordinati in modo crescente rispetto al cognome (prima) e al nome (poi)*

SQL:

```
select * from segreteria.studenti
      where anno_corso = 1 or anno_corso =2
      order by data_nascita, cognome, nome;
```

SQLAlchemy:

```
select(studenti).\
   where((studenti.c.anno_corso == 1) | (studenti.c.anno_corso == 2)).\
      order_by(studenti.c.data_nascita, studenti.c.cognome, studenti.c.nome)
```

Seconda parte

2.a) *matricola degli studenti che hanno superato l'esame di Programmazione con un voto superiore a 27*

SQL:

```
select studente from segreteria.esami, segreteria.corsi
        where corso = cod_corso and voto > 27 and nome = 'Programmazione';

select studente from segreteria.esami join segreteria.corsi
        on corso = cod_corso where voto > 27 and nome = 'Programmazione';
```

SQLAlchemy:

```
select(esami.c.studente).join(corsi, esami.c.corso == corsi.c.cod_corso).\
    where((esami.c.voto > 27) & (corsi.c.nome ==  'Programmazione'))
```

2.b) *cognome e nome degli studenti che hanno superato almeno un esame nel 2007, con l'eliminazione di eventuali duplicati*

SQL:

```
select distinct cognome, nome from segreteria.studenti, segreteria.esami
    where matricola = studente and data between '2007-01-01' and '2007-12-31';

select distinct cognome, nome from segreteria.studenti join segreteria.esami
    on matricola = studente where data between '2007-01-01' and '2007-12-31';
```

SQLAlchemy:

```
select(studenti.c.cognome, studenti.c.nome).\
    join(esami, esami.c.studente == studenti.c.matricola).\
        where(esami.c.data.between('2007-01-01', '2007-12-31')).distinct()
```

2.c) *dati relativi agli esami sostenuti dallo studente Mario Rossi, ordinati in senso decrescente rispetto al voto e, a parità di voto, in senso crescente rispetto alla data*

SQL:

```
select segreteria.esami.* from segreteria.esami, segreteria.studenti
        where studente = matricola
        and cognome = 'Rossi' and nome = 'Mario'
        order by voto desc, data;

select segreteria.esami.* from segreteria.esami join segreteria.studenti
        on studente = matricola
        where cognome = 'Rossi' and nome = 'Mario'
        order by voto desc, data;
```

SQLAlchemy:

```
select(esami).\
    join(studenti, esami.c.studente == studenti.c.matricola).\
        where((studenti.c.cognome == 'Rossi') & (studenti.c.nome == 'Mario')).\
            order_by(desc(esami.c.voto), esami.c.data)
```

2.d) *coppie (matricola1, matricola2) di studenti omonimi (stesso cognome e stesso nome)*

SQL:

```
select S1.matricola, S2.matricola from segreteria.studenti S1, segreteria.studenti S2
                    where S1.cognome = S2.cognome and S1.nome = S2.nome
                    and S1.matricola <> S2.matricola;

select S1.matricola, S2.matricola from segreteria.studenti S1
                    join segreteria.studenti S2 on (S1.cognome = S2.cognome
                    and S1.nome = S2.nome and S1.matricola <> S2.matricola);
```

SQLAlchemy:

```
s1, s2 = studenti.alias(), studenti.alias()
select(s1.c.matricola.label('matricola1'), s2.c.matricola.label('matricola2')).\
    where((s1.c.cognome == s2.c.cognome) & \
        (s1.c.nome ==s2.c.nome)&(s1.c.matricola !=s2.c.matricola))
```

2.e) *dati degli studenti del primo anno con il codice e il voto dei relativi esami sostenuti, inclusi gli studenti che non hanno sostenuto alcun esame*

SQL:

```
select segreteria.studenti.*, corso, voto
    from segreteria.studenti left join segreteria.esami
        on matricola = studente where anno_corso = 1;
```

SQLAlchemy:

```
select(studenti, esami.c.corso, esami.c.voto).\
    join(esami, studenti.c.matricola == esami.c.studente, isouter = True).\
        where(studenti.c.anno_corso ==1 )
```

2.f) *dati di tutti i docenti con i relativi insegnamenti, inclusi i docenti che non tengono alcun corso*

SQL :

```
select D.*, C.cod_corso, C.nome
    from segreteria.docenti D left join segreteria.corsi C
        on D.cod_docente = C.docente;
```

SQLAlchemy:

```
select(docenti, corsi.c.cod_corso, corsi.c.nome).\
    join(corsi, docenti.c.cod_docente == corsi.c.docente, isouter = True)
```