Assessment 1

Database Management Systems

Sreehari P Sreedhar CB.SC.I5DAS20032

In [ ]:
```
import os

try:
    os.remove('../Dumps/Assessment1.db')
except FileNotFoundError:
    pass
```

In [ ]:
```
import sqlite3
```

In [ ]:
```
conn = sqlite3.connect('../Dumps/Assessment1.db')

cur = conn.cursor()
```

In [ ]:
```
createTables = '''
    BEGIN;

    CREATE TABLE IF NOT EXISTS SALESMAN (
        SALESMAN_ID INTEGER PRIMARY KEY,
        NAME VARCHAR(25),
        CITY VARCHAR(25),
        COMMISSION INTEGER(2, 2)
        );

    CREATE TABLE IF NOT EXISTS CUSTOMER (
        CUSTOMER_ID INTEGER PRIMARY KEY,
        CUST_NAME VARCHAR(25),
        CITY VARCHAR(25),
        GRADE INTEGER,
        SALESMAN_ID INTEGER REFERENCES SALESMAN(SALESMAN_ID) ON UPDATE CASCADE
        );
```

```
    CREATE TABLE IF NOT EXISTS ORDERS (
        ORD_NO INTEGER PRIMARY KEY,
        PURCH_AMT INTEGER(5, 2),
        ORD_DATE DATE,
        CUSTOMER_ID INTEGER REFERENCES CUSTOMER(CUSTOMER_ID) ON UPDATE CASCADE,
        SALESMAN_ID INTEGER REFERENCES SALESMAN(SALESMAN_ID) ON UPDATE CASCADE
        );

    COMMIT;
'''
```

In [ ]:
```
cur.executescript(createTables)
```

Out[ ]:
```
<sqlite3.Cursor at 0x17f807fbc40>
```

In [ ]:
```
salesMen = [
    (1001, 'zavier', 'new york', 0.11),
    (1002, 'zenpaul', 'denmark', 0.23),
    (1003, 'alen', 'chicago', 0.45),
    (1004, 'boby', 'new delhi', 0.33),
    (1005, 'cheran', 'denmark', 0.24),
    (1006, 'alex', 'new york', 0.21)
]

cur.executemany('INSERT INTO SALESMAN VALUES (?, ?, ?, ?)', salesMen)

conn.commit();
```

In [ ]:
```
customers = [
    (2001, 'nicklado', 'new york', 100, 1001),
    (2002, 'ritwik', 'denmark', 200, 1003),
    (2003, 'sachin', 'chicago', 300, 1002),
    (2004, 'dora', 'new delhi', 100, 1004),
    (2005, 'alwin', 'denmark', 100, 1005),
    (2006, 'simon', 'chicago', 200, 1002),
    (2007, 'rohan', 'pitsberg', 200, 1003),
    (2008, 'aswin', 'pitsberg', 300, 1006)
]

cur.executemany('INSERT INTO CUSTOMER VALUES (?, ?, ?, ?, ?)', customers)
```

```
conn.commit()
```

In [ ]:
```
orders = [
    (1, 1000, '2012-02-01', 2001, 1001),
    (2, 2000, '2012-01-01', 2002, 1003),
    (3, 3000, '2012-03-01', 2003, 1002),
    (4, 4000, '2012-07-01', 2004, 1004),
    (5, 5000, '2012-12-01', 2005, 1005),
    (6, 1000, '2012-03-01', 2006, 1006),
    (7, 2500, '2012-11-01', 2007, 1003),
    (8, 3000, '2012-09-01', 2008, 1006),
    (9, 5000, '2012-02-02', 2001, 1001),
    (10, 5000, '2012-12-15', 2001, 1001)
]

cur.executemany('INSERT INTO ORDERS VALUES (?, ?, ?, ?, ?)', orders)

conn.commit()
```

Questions:

1. Identify the Purchase_amount between 2000 to 5000 and print the ord_no, purchase_amount and customer city.

In [ ]:
```
cur.execute('''
    SELECT O.ORD_NO, O.PURCH_AMT, C.CITY
    FROM ORDERS O INNER JOIN CUSTOMER C ON O.CUSTOMER_ID = C.CUSTOMER_ID
    WHERE (O.PURCH_AMT >= 2000 AND O.PURCH_AMT <= 5000);
    ''').fetchall()
```

Out[ ]:
```
[(2, 2000, 'denmark'),
 (3, 3000, 'chicago'),
 (4, 4000, 'new delhi'),
 (5, 5000, 'denmark'),
 (7, 2500, 'pitsberg'),
 (8, 3000, 'pitsberg'),
 (9, 5000, 'new york'),
 (10, 5000, 'new york')]
```

2. Find the commission associated to each customer and salesman and display customer name, salesman name, commission percentile.

In [ ]:
```
cur.execute('''
    SELECT C.CUST_NAME, S.NAME, S.COMMISSION
    FROM CUSTOMER C INNER JOIN SALESMAN S ON C.SALESMAN_ID = S.SALESMAN_ID;
''').fetchall()
```

Out[ ]:
```
[('nicklado', 'zavier', 0.11),
 ('ritwik', 'alen', 0.45),
 ('sachin', 'zenpaul', 0.23),
 ('dora', 'boby', 0.33),
 ('alwin', 'cheran', 0.24),
 ('simon', 'zenpaul', 0.23),
 ('rohan', 'alen', 0.45),
 ('aswin', 'alex', 0.21)]
```

3. Identify the salesman who fixed commission above 20% and display salesman name, customer name, commission.

In [ ]:
```
cur.execute('''
    SELECT C.CUST_NAME, S.NAME, S.COMMISSION
    FROM CUSTOMER C INNER JOIN SALESMAN S ON C.SALESMAN_ID = S.SALESMAN_ID
    WHERE S.COMMISSION > 0.2;
''').fetchall()
```

Out[ ]:
```
[('ritwik', 'alen', 0.45),
 ('sachin', 'zenpaul', 0.23),
 ('dora', 'boby', 0.33),
 ('alwin', 'cheran', 0.24),
 ('simon', 'zenpaul', 0.23),
 ('rohan', 'alen', 0.45),
 ('aswin', 'alex', 0.21)]
```

4. Display the customer name, customer city, salesman name , salesman city where condition is salesman city != customer city.

In [ ]:
```
cur.execute('''
    SELECT C.CUST_NAME, C.CITY, S.NAME, S.CITY
    FROM CUSTOMER C INNER JOIN SALESMAN S ON C.SALESMAN_ID = S.SALESMAN_ID
    WHERE C.CITY != S.CITY;
''').fetchall()
```

Out[ ]:
```
[('ritwik', 'denmark', 'alen', 'chicago'),
 ('sachin', 'chicago', 'zenpaul', 'denmark'),
 ('simon', 'chicago', 'zenpaul', 'denmark'),
 ('rohan', 'pitsberg', 'alen', 'chicago'),
 ('aswin', 'pitsberg', 'alex', 'new york')]
```

5. Display the following information of all the orders like order number, order date, purchase_amt, Customer Name, Salesman name.

In [ ]:
```
cur.execute('''
    SELECT O.ORD_NO, O.ORD_DATE, O.PURCH_AMT, C.CUST_NAME, S.NAME
    FROM ORDERS O INNER JOIN (
        CUSTOMER C INNER JOIN SALESMAN S ON C.SALESMAN_ID = S.SALESMAN_ID
    ) ON O.CUSTOMER_ID = C.CUSTOMER_ID
''').fetchall()
```

Out[ ]:
```
[(1, '2012-02-01', 1000, 'nicklado', 'zavier'),
 (2, '2012-01-01', 2000, 'ritwik', 'alen'),
 (3, '2012-03-01', 3000, 'sachin', 'zenpaul'),
 (4, '2012-07-01', 4000, 'dora', 'boby'),
 (5, '2012-12-01', 5000, 'alwin', 'cheran'),
 (6, '2012-03-01', 1000, 'simon', 'zenpaul'),
 (7, '2012-11-01', 2500, 'rohan', 'alen'),
 (8, '2012-09-01', 3000, 'aswin', 'alex'),
 (9, '2012-02-02', 5000, 'nicklado', 'zavier'),
 (10, '2012-12-15', 5000, 'nicklado', 'zavier')]
```

6. Join all the 3 tables and display the complete information from all tables in such a way that the columns should not repeat.

In [ ]:
```
cur.execute('''
    SELECT S.SALESMAN_ID, S.NAME, S.CITY, S.COMMISSION, C.CUSTOMER_ID, C.CUST_NAME, C.CITY, C.GRADE, O.ORD_NO, O.ORD_DATE, O.PURC
    FROM ORDERS O INNER JOIN (
        CUSTOMER C INNER JOIN SALESMAN S ON C.SALESMAN_ID = S.SALESMAN_ID
    ) ON O.CUSTOMER_ID = C.CUSTOMER_ID;
''').fetchall()
```

Out[ ]:
```
[(1001,
  'zavier',
  'new york',
  0.11,
  2001,
  'nicklado',
  'new york',
  100,
  1,
  '2012-02-01',
  1000),
 (1003,
  'alen',
  'chicago',
  0.45,
  2002,
  'ritwik',
  'denmark',
  200,
  2,
  '2012-01-01',
  2000),
 (1002,
  'zenpaul',
  'denmark',
  0.23,
  2003,
  'sachin',
  'chicago',
  300,
  3,
  '2012-03-01',
  3000),
 (1004,
  'boby',
  'new delhi',
  0.33,
  2004,
  'dora',
  'new delhi',
  100,
```

```
 4,
 '2012-07-01',
 4000),
(1005,
 'cheran',
 'denmark',
 0.24,
 2005,
 'alwin',
 'denmark',
 100,
 5,
 '2012-12-01',
 5000),
(1002,
 'zenpaul',
 'denmark',
 0.23,
 2006,
 'simon',
 'chicago',
 200,
 6,
 '2012-03-01',
 1000),
(1003,
 'alen',
 'chicago',
 0.45,
 2007,
 'rohan',
 'pitsberg',
 200,
 7,
 '2012-11-01',
 2500),
(1006,
 'alex',
 'new york',
 0.21,
 2008,
 'aswin',
```

```
 'pitsberg',
 300,
 8,
 '2012-09-01',
 3000),
(1001,
 'zavier',
 'new york',
 0.11,
 2001,
 'nicklado',
 'new york',
 100,
 9,
 '2012-02-02',
 5000),
(1001,
 'zavier',
 'new york',
 0.11,
 2001,
 'nicklado',
 'new york',
 100,
 10,
 '2012-12-15',
 5000)]
```

7. Sort the table based on the Customer_id in ascending. Display customer name, customer city, grade, salesman, salesman city.

In [ ]:
```
cur.execute('''
    SELECT C.CUSTOMER_ID, C.CUST_NAME, C.CITY, C.GRADE, S.SALESMAN_ID, S.NAME, S.CITY
    FROM CUSTOMER C INNER JOIN SALESMAN S ON C.SALESMAN_ID = S.SALESMAN_ID
    ORDER BY C.CUSTOMER_ID ASC;
''').fetchall()
```

Out[ ]:
```
[(2001, 'nicklado', 'new york', 100, 1001, 'zavier', 'new york'),
 (2002, 'ritwik', 'denmark', 200, 1003, 'alen', 'chicago'),
 (2003, 'sachin', 'chicago', 300, 1002, 'zenpaul', 'denmark'),
 (2004, 'dora', 'new delhi', 100, 1004, 'boby', 'new delhi'),
 (2005, 'alwin', 'denmark', 100, 1005, 'cheran', 'denmark'),
 (2006, 'simon', 'chicago', 200, 1002, 'zenpaul', 'denmark'),
 (2007, 'rohan', 'pitsberg', 200, 1003, 'alen', 'chicago'),
 (2008, 'aswin', 'pitsberg', 300, 1006, 'alex', 'new york')]
```

8. Display the information of salesperson who worked with more than one customer. Fetch the Salesman name, salesman city customer name customer city.

In [ ]:
```
cur.execute('''
    SELECT C.CUST_NAME, C.CITY, S.NAME, S.CITY, CNT.VALUE
    FROM SALESMAN S INNER JOIN (
        (SELECT COUNT(C.SALESMAN_ID) AS VALUE, C.SALESMAN_ID
        FROM CUSTOMER C
        GROUP BY C.SALESMAN_ID) AS CNT INNER JOIN CUSTOMER C ON CNT.SALESMAN_ID = C.SALESMAN_ID
    ) ON C.SALESMAN_ID = S.SALESMAN_ID
    WHERE CNT.VALUE > 1;
''').fetchall()
```

Out[ ]:
```
[('ritwik', 'denmark', 'alen', 'chicago', 2),
 ('sachin', 'chicago', 'zenpaul', 'denmark', 2),
 ('simon', 'chicago', 'zenpaul', 'denmark', 2),
 ('rohan', 'pitsberg', 'alen', 'chicago', 2)]
```

9. Update the commission of all rows by 0.20. Eg: .011+.20=0.31.

In [ ]:
```
cur.execute('SELECT * FROM SALESMAN;').fetchall()
```

Out[ ]:
```
[(1001, 'zavier', 'new york', 0.11),
 (1002, 'zenpaul', 'denmark', 0.23),
 (1003, 'alen', 'chicago', 0.45),
 (1004, 'boby', 'new delhi', 0.33),
 (1005, 'cheran', 'denmark', 0.24),
 (1006, 'alex', 'new york', 0.21)]
```

In [ ]:
```
cur.execute('''
```

```
    UPDATE SALESMAN SET COMMISSION = COMMISSION + 0.2;
''').fetchall()
```

Out[ ]:
```
[]
```

In [ ]:
```
cur.execute('SELECT * FROM SALESMAN;').fetchall()
```

Out[ ]:
```
[(1001, 'zavier', 'new york', 0.31),
 (1002, 'zenpaul', 'denmark', 0.43000000000000005),
 (1003, 'alen', 'chicago', 0.65),
 (1004, 'boby', 'new delhi', 0.53),
 (1005, 'cheran', 'denmark', 0.44),
 (1006, 'alex', 'new york', 0.41000000000000003)]
```

10. Modify the column name CUSTOMER_ID and SALESMAN_ID as varchar(25). Accept few varchar values to the appropriate tables.

Standard SQL command to change datatype of column

```
MODIFY COLUMN SALESMAN.SALESMAN_ID VARCHAR(25);
MODIFY COLUMN CUSTOMER.CUSTOMER_ID VARCHAR(25);
```

In [ ]:
```
conn.close()
```