Assignment 2 [part 1 of 2]

Questions 1-2 are pen-and-paper exercises; questions 3-4 are coding exercises. For all your answers to the assignment, please include your workings (e.g. equations, code) when this is relevant to the question.

1). A data warehouse for a music streaming company consists of the dimensions song, user, time (time and date of when the user listened to a song), and the two measures count (how many times a user listened to the song) and fee (fee paid by the streaming company to the artist every time a user listens to that song).

1. Draw a schema diagram for the above data warehouse using a star schema. [1 mark out of 5]
2. Starting with the base cuboid [time, user, song], what specific OLAP operations should be performed in order to list the total fee collected for a given song for a given month of a given year (e.g. October 2021)? [0.5 marks out of 5]
3. Assume that the time dimension has 4 levels: day, month, quarter, year; and that the song and user dimensions both have 1 level (not including the virtual level 'all'). How many cuboids will this cube contain (including the base and apex cuboids)? [0.5 marks out of 5]

2). Suppose that a car rental company has a data warehouse that holds record ID lists of vehicles in terms of brands (Audi, Ford, Mini) and store branches (Tower Hamlets, Newham, Hackney). Each record consists of a combination of vehicle brand and branch, and records for all combinations exist. We would like to index the OLAP data using bitmap indices. Write down the *base table* for record IDs, and the corresponding *bitmap index table* for vehicle brand. [0.5 marks out of 5]

3). Using the same CSV file and data cube in the above lab tutorial, modify the "tutorial_model.json" file to include aggregate measures for the minimum and maximum amount in the data cube. Using these implemented aggregate measures, produce the values for the minimum and maximum amount in the data per year. Make sure to show your workings in the PDF report. You can read the Cubes package documentation for assistance in this task. [1 mark out of 5]

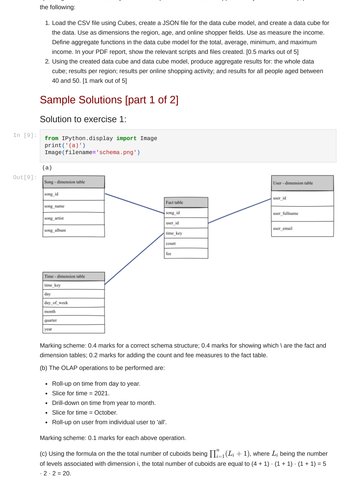
4). Using the CSV file "country-income.csv" (found in the week 5 supplementary lab documents), perform the following:

1. Load the CSV file using Cubes, create a JSON file for the data cube model, and create a data cube for the data. Use as dimensions the region, age, and online shopper fields. Use as measure the income. Define aggregate functions in the data cube model for the total, average, minimum, and maximum *income*. In your PDF report, show the relevant scripts and files created. [0.5 marks out of 5]
2. Using the created data cube and data cube model, produce aggregate results for: the whole data cube; results per region; results per online shopping activity; and results for all people aged between 40 and 50. [1 mark out of 5]

Sample Solutions [part 1 of 2]

Solution to exercise 1:

In [0]:
```
from IPython.display import Image
print('{a}')
Image(filename='schema.png')
```
{a}

Out[0]:


Marking scheme: 0.4 marks for a correct schema structure; 0.4 marks for showing which \ are the fact and dimension tables; 0.2 marks for adding the count and fee measures to the fact table.

(b) The OLAP operations to be performed are:

- Roll-up on time from day to year.
- Slice for time = 2021.
- Drill-down on time from year to month.
- Slice for time = October.
- Roll-up on user from individual user to 'all'.

Marking scheme: 0.1 marks for each above operation.

(c) Using the formula on the the total number of cuboids being $\prod_{i=1}^{n}(L_i + 1)$, where L_i being the number of levels associated with dimension i, the total number of cuboids are equal to (4 + 1) · (1 + 1) · (1 + 1) = 5 · 2 · 2 = 20.

Marking scheme: 0.3 marks for correct result; 0.2 marks for students showing their workings.

Solution to exercise 2:

In [0]:
```
Image(filename='table.png')
```

Out[0]:
Base Table *(0.25 marks)*

RID	Brand	Branch
R1	Audi	Tower Hamlets
R2	Audi	Newham
R3	Audi	Hackney
R4	Ford	Tower Hamlets
R5	Ford	Newham
R6	Ford	Hackney
R7	Mini	Tower Hamlets
R8	Mini	Newham
R9	Mini	Hackney

Index on Brand *(0.25 marks)*

RID	Audi	Ford	Mini
R1	1	0	0
R2	1	0	0
R3	1	0	0
R4	0	1	0
R5	0	1	0
R6	0	1	0
R7	0	0	1
R8	0	0	1
R9	0	0	1

Solution to exercise 3:

In [12]:
```
# marking scheme:
# 0.5 marks for the modified JSON file
# 0.5 marks for producing the implemented aggregate measures

# You can create a new model file (e.g. tutorial_model_modified.json) and include the
# under the "aggregates" field:
#                 {
#                     "name": "amount_min",
#                     "function": "min",
#                     "measure": "amount"
#                 },
#                 {
#                     "name": "amount_max",
#                     "function": "max",
#                     "measure": "amount"
#                 }
from cubes import Workspace
workspace = Workspace()
workspace.register_default_store("sql", url="sqlite:///data.sqlite")
workspace.import_model("tutorial_model.json")
browser = workspace.browser("ibrd_balance")
result = browser.aggregate(drilldown=["year"])
for record in result:
    print(record)
```

```
{'year': 2009, 'amount_min': -1683, 'amount_max': 118040, 'amount_sum': 550840, 'record_count': 31}
{'year': 2010, 'amount_min': -3043, 'amount_max': 128577, 'amount_sum': 566020, 'record_count': 31}
```

Solution to exercise 4:

In [18]:
```
import cubes as cubes
from sqlalchemy import create_engine
from cubes.tutorial.sql import create_table_from_csv
# data cube creation - 0.25 marks
engine = create_engine('sqlite:///data.sqlite')
create_table_from_csv(engine,
                      "country-income.csv",
                      table_name="country-income",
                      fields=[
                          ("region", "string"),
                          ("age", "integer"),
                          ("income", "integer"),
                          ("online_shopper", "string")],
                      create_id=True)

workspace = Workspace()
workspace.register_default_store("sql", url="sqlite:///data.sqlite")
workspace.import_model("income_model.json") # 0.25 marks for the created JSON file
cube_income = workspace.cube('country-income')
browser_income = workspace.browser(cube_income)


# print aggregate results for the entire data cube - 0.25 marks
result = browser_income.aggregate()
print(result.summary)

# print aggregate results per region - 0.25 marks
result = browser_income.aggregate(drilldown=["region"])
for record in result:
    print(record)

# print aggregate results per online shopper field - 0.25 marks
result = browser_income.aggregate(drilldown=["online_shopper"])
for record in result:
    print(record)

# print aggregate results for people with age between 40 and 50 - 0.25 marks
cuts = [cubes.RangeCut("age", ["40"], ["50"])]
cell = cubes.Cell(cube_income, cuts)
result = browser_income.aggregate(cell)
result.summary
```

```
{'income_sum': 768200, 'income_avg': 76820.0, 'income_min': 57600, 'income_max': 99600}
{'region': 'Brazil', 'income_sum': 193200, 'income_avg': 64400.0, 'income_min': 57600, 'income_max': 71600}
{'region': 'India', 'income_sum': 331200, 'income_avg': 82800.0, 'income_min': 69600, 'income_max': 94800}
{'region': 'USA', 'income_sum': 243800, 'income_avg': 81266.66666666667, 'income_min': 64800, 'income_max': 97600}
{'online_shopper': 'No', 'income_sum': 386400, 'income_avg': 77280.0, 'income_min': 62400, 'income_max': 99600}
{'online_shopper': 'Yes', 'income_sum': 381800, 'income_avg': 76360.0, 'income_min': 57600, 'income_max': 94800}
{'income_sum': 451400,
 'income_avg': 75233.33333333333,
 'income_min': 62400,
 'income_max': 86400}
```

Assignment 2 [part 2 of 2]

For your answers to the assignment, please include include your workings (e.g. equations, code) when this is relevant to the question. Questions 1-5 are pen-and-paper exercises. Questions 6 and 7 below require you to write and provide code.

1. Consider a dataset $\mathcal{D}$ that contains only two observations $\mathbf{x}_1 = (1, -1)$ and $\mathbf{x}_2 = (-1, 1)$. Suppose that the class of the first observation is $y_1 = 0$ and that the class of the second observation is $y_2 = 1$. How would a 1-nearest neighbour classifier based on the Euclidean distance classify the observation $\mathbf{x} = (-2, 3)$? What are the distances between this new observation and each observation in the dataset? [0.5 marks out of 5]
2. Consider a dataset $\mathcal{D}$ that only contains observations of two different classes. Explain why a k-nearest neighbour classifier does not need a tie-breaking policy when k is odd. [0.5 marks out of 5]
3. Explain why a classifier that obtains an accuracy of 99.9% can be terrible for some datasets. [0.5 marks out of 5]
4. Consider a classifier tasked with predicting whether an observation belongs to class y (positive class). Suppose that this classifier has precision 1.0 and recall 0.1 on a test dataset. If this classifier predicts that an observation does not belong to class y, should it be trusted? Should it be trusted if it predicts that the observation belongs to class y? [0.5 marks out of 5]
5. Based on the confusion matrix shown in this lab notebook, what is the pair of classes that is most confusing for the 1-nearest neighbour classifier trained in the previous sections? [0.5 marks out of 5]
6. Train a support vector machine classifier using the same (subsampled) training dataset used in the previous sections and compute its accuracy on the corresponding test dataset. You can use the default hyperparameters for the class `SVC` from `sklearn.svm`. Show the code in the report. [1.25 marks out of 5]
7. Using the same (subsampled) training dataset used in the previous sections, employ `GridSearchCV` to find the best hyperparameter settings based on 5-fold cross-validation for a `RandomForestClassifier`. Consider `n_estimators` $\in \{50, 100, 200\}$ and `max_features` $\in \{0.1, 0.25\}$. Use the default values for the remaining hyperparameters. Compute the accuracy of the best model on the corresponding test dataset. Show the code in the report. [1.25 marks out of 5]

Sample Solutions [part 2 of 2]

Solution to exercises 1-5 (brief)

1. Distances: $e(\mathbf{x}_1, \mathbf{x}) = 5$, $e(\mathbf{x}_2, \mathbf{x}) = 2.23$ *(0.25 marks)*. Therefore, the observation $\mathbf{x}$ would be assigned to class 1 *(0.25 marks)*.
2. The votes for the two different classes must add up to an odd number k. A tie happens when both classes receive the same number of votes. In that case, the votes would have to add up to an even number. Therefore, a tie cannot happen when k is odd. *(0.5 marks)*
3. If 99.9% of the observations in a test dataset belong to one of the classes, a classifier that always predicts that class will have a test accuracy of exactly 99.9%. *(0.5 marks)*
4. Should not be trusted when predicts not y *(0.25 marks)*. Should be trusted when predicts y *(0.25 marks)*.
5. 4 and 9. *(0.5 marks)*

Solution to exercise 6

In [23]:
```
# marking scheme:
# 0.25 marks for loading and using the same subsampled dataset from previous sections
# 0.5 marks for training the SVM classifier
# 0.5 marks for displaying the test dataset accuracy

# Configuring the appearance of ``seaborn`` graphics in this notebook
import grip
import pickle
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

f = grip.open('Supplementary_Material_Lab05/data/mnist.pkl.gz', 'rb')
X, y = pickle.load(f, encoding='latin1')[0]
f.close()

# Subsampling
sample_size = 2000
X, y = X[:sample_size], y[:sample_size]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)


svm = SVC()
svm.fit(X_train, y_train)
print('Test dataset accuracy: {0:.}'.format(svm.score(X_test, y_test)))
```

```
Test dataset accuracy: 0.9275.
```

Solution to exercise 7

In [24]:
```
# marking scheme:
# 0.25 marks for loading and using the same subsampled dataset from previous sections
# 0.5 marks for correctly training a random forest classifier and correctly setting the
# 0.5 marks for displaying the best hyperparameter settings and for showing the corre

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV

f = grip.open('Supplementary_Material_Lab05/data/mnist.pkl.gz', 'rb')
X, y = pickle.load(f, encoding='latin1')[0]
f.close()

# Subsampling
sample_size = 2000
X, y = X[:sample_size], y[:sample_size]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

hyperparameters = {
    'n_estimators': [50, 100, 200],
    'max_features': [0.1, 0.25],
}

rfc = RandomForestClassifier(random_state=0)
rfc_cv = GridSearchCV(rfc, hyperparameters, cv=5)
rfc_cv.fit(X_train, y_train)

print('Best hyperparameter setting: {0}.'.format(rfc_cv.best_estimator_))
print('Average accuracy across folds of best hyperparameter setting: {0}.'.format(rfc_cv.
print('Test dataset accuracy of best hyperparameter setting: {0}.'.format(rfc_cv.score

Best hyperparameter setting: RandomForestClassifier(max_features=0.1, n_estimators=20
0, random_state=0).
Average accuracy across folds of best hyperparameter setting: 0.9106249999999999.
Test dataset accuracy of best hyperparameter setting: 0.915.
```

In [ ]: