MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Introduction to the MIMIC II database

1. Overview

This tutorial provides an introduction to the database structure and content. It also provides an idea of the types of information which can be extracted and the complexity of the data contained. By the end of this tutorial you will be able to:

- Obtain meta-data from the various database objects (Tables, views, etc).
- Perform basic queries on a single table. This includes counting the number of rows, and restricting the query to a subset of rows.
- Perform basic 'joins' to combine tables and extract useful information.
- Use database 'views' to extract high-level information.

Commence the tutorial by opening the **QueryBuilder** application.

2. Database meta-data

The meta-data for a particular table can be obtained by clicking on an entry on the left hand side of the screen. Select the d_patients table to see how the metadata is displayed in the panels on the right hand side of the screen. The columns found in the d_patients table are displayed, along with the types of data they contain, and various other parameters. Also provided are comments which describe the data contained in the columns.

Try selecting some other tables and look at the metadata. You can close a tab by clicking on the 'X' on the metadata tabs. When you have finished, close all of the metadata tabs and go to No. 3

3. Patient Numbers

Ensure that the 'Query...' tab at the top of the screen is selected. SQL queries can be entered in the top panel and the results will be displayed at the bottom when the 'Execute query' button is pressed. Enter the following SQL in the box and press the 'Execute query' button.

```
SELECT * FROM d_patients
```

At the bottom of the screen you will see three columns: subject_id, sex, and date of birth. 50 records are retrieved at a time and you can page through the results using the controls at the bottom of the screen.

Obtain the number of patients by performing the following query:

```
SELECT COUNT(*) FROM d_patients
```

The 'sex' column identifies the gender of the patient. We can obtain the values used to indicate patient genders using the following query:

```
SELECT DISTINCT sex FROM d_patients
```

We can see that 'M' and 'F' are the two characters used to indicate patient sex. We can use this information to obtain the number of female patients by restricting the query to retrieve results which have 'F' in the 'sex' column:

```
SELECT COUNT(*) FROM d_patients WHERE sex = 'F'
```

And the numbers of male and female patients can be obtained using this query:

```
SELECT sex,
       COUNT(*)
FROM d_patients
GROUP BY sex
```

4. Mortality and admissions

A flag which records whether or not a patient died in the hospital is stored in the d_patients table. Count the number of patients who died using the following query:

```
SELECT hospital_expire_flg,
 COUNT(*)
FROM d_patients GROUP BY hospital_expire_flg
```

The database also contains date of death for patients who died outside the hospital, based on social security death records. This information is contained in the 'dod' column. Please note that this database contains adult, pediatric and neonatal patients which will affect the mortality statistics. Categorizing patients into different age groups is carried out in the next section.

5. Patient age and mortality

To determine the adult mortality rate, we must first determine adult patients. We define adults as those patients who are 15 or more years old at the date of their first admission. To perform this query, we must first combine the d_patients and admissions tables to find patient admission dates, and their date of birth. Please note the table naming in the query below. We have denoted 'admissions' with 'a' and 'd_patients' with 'p':

```
SELECT p.subject_id, p.dob,
 a.hadm_id, a.admit_dt, p.hospital_expire_flg
FROM admissions a,
 d_patients p
WHERE p.subject_id = a.subject_id
```

Next, we find the minimum (earliest) admission date for each patient. This requires the use of new functions, the 'MIN' function, which obtains the minimum value, and the 'PARTITION BY' function which determines the groups over which the minimum value is obtained, in this case, we determine the minimum time of admission for each patient:

```
SELECT DISTINCT p.subject_id, p.dob,
 a.hadm_id, a.admit_dt, p.hospital_expire_flg,
 MIN(a.admit_dt)
 OVER(PARTITION BY p.subject_id)
 AS first_adm_dt
 FROM admissions a,
 d_patients p
 WHERE p.subject_id = a.subject_id
 AND p.dob IS NOT NULL
 ORDER BY a.hadm_id, p.subject_id
```

A patient's age is given by the difference between their date of birth and the date of their first admission. We can obtain this by combining the above query with another query to provide the ages. Furthermore, we assign categories to different ages: >=15 years old are adults and the rest are assigned to the 'other' category. The queries are combined using the

'WITH' keyword:

```
WITH first_admission_date AS (
 SELECT DISTINCT p.subject_id, p.dob, p.sex,
 a.hadm_id, a.admit_dt,
 MIN(a.admit_dt)
 OVER(PARTITION BY a.hadm_id, p.subject_id)
 AS first_adm_dt
 FROM admissions a,
 d_patients p
 WHERE p.subject_id = a.subject_id
 AND p.dob IS NOT NULL
 ORDER BY a.hadm_id, p.subject_id
),
age AS (
 SELECT subject_id, hadm_id, dob, sex,
 first_adm_dt,
 ROUND(months_between(first_adm_dt, dob) /12, 2) first_adm_age,
 CASE
 WHEN (months_between(first_adm_dt, dob) /12) >= 15
 THEN 'adult'
 WHEN months_between(first_adm_dt, dob) <= 1
 THEN 'neonate'
 ELSE 'middle'
 END AS age_group
 FROM first_admission_date
 ORDER BY subject_id, hadm_id
)
SELECT * FROM age
```

The above query can now be combined with the **WHERE** and **COUNT** functions described earlier to determine the number of adult patients, whether or not they died, and therefore, their mortality rate.

6. ICU Stays

In the MIMIC-II database, we define an ICU stay to be continuous if a patient is returned to an ICU room within 24 hrs of being moved to a ward. Patient ICU movements are recorded in the censusevents table:

```
SELECT * FROM censusevents
```

The columns should be fairly self explanatory, click on the censusevents table on the left hand side if you need more information about the columns and the data they contain. The 'careunit' and 'destcareunit' contain the IDs of the originating, and destination careunit respectively. Their data is contained within the 'd_careunits' table which can be joined to 'censusevents' to see the care unit names:

```
SELECT ce.census_id, ce.subject_id, ce.intime, ce.outtime,
ce.careunit, cu.label, ce.destcareunit, dcu.label destunitname,
ce.dischstatus, ce.los
FROM censusevents ce, d_careunits cu, d_careunits dcu
WHERE ce.careunit = cu.cuid
AND ce.destcareunit = dcu.cuid
```

The information in the 'censusevents' table has been combined with the data in the d_patients, admissions, d_chartitems and chartevents table to create a 'view'. Views can be thought of as saved queries which contain useful information which is often requested from the database. As we determine common queries which users frequently perform, we will add more views to the database to provide simple access to useful data. The 'icustay_detail' is one such view which has been added to the database. Look at the information contained in the 'icustay_detail' view now by displaying its contents:

```
SELECT * FROM icustay_detail
```

As you can see, this view contains a large amount of detail related to patient ICU stays which has been extracted from other tables. The patients' gender and date of birth are supplied from the d_patients table. The patient age at admission has been used to categorize them into 'adult', 'neonate' and 'other'. Hospital admission and ICU admission information has been obtained from the admissions and censusevents table. Each admission has been assigned a number and the total number of hospital and ICU admissions for each patient has been calculated. The expire_flg columns for hospital and ICU stays has been provided to determine mortality and the first and last service type of each stay is shown too.

This view of the database allows the mortality statistics calculated earlier to be obtained more easily. For example, adult patients can be obtained with:

```
SELECT * FROM icustay_detail WHERE icustay_age_group = 'adult'
```

Adult patient mortality (number of deaths/number of ICU stays) can be obtained with

```
SELECT icustay_expire_flg, count(icustay_expire_flg)
FROM icustay_detail
WHERE icustay_age_group = 'adult'
GROUP BY icustay_expire_flg
```

7. Problem

How many patients have at least one mean arterial pressure measurement of less than 60 mmHg and what is their mortality rate?

Hint: the vital_signs_raw table contains MAP readings and mortality flags.