Assignment 3 SQL query by Snowflake

Student name: Phan Hoang An Nguyen

Student ID: 100404103

Database

Dataset name: eICU Collaborative Research Database v2.0

Dataset source: The eICU Collaborative Research Database, a freely available multi-center database for critical care research. Pollard TJ, Johnson AEW, Raffa JD, Celi LA, Mark RG and Badawi O. Scientific Data (2018). DOI: 10.1038/sdata.2018.178. Available at: https://www.nature.com/articles/sdata2018178

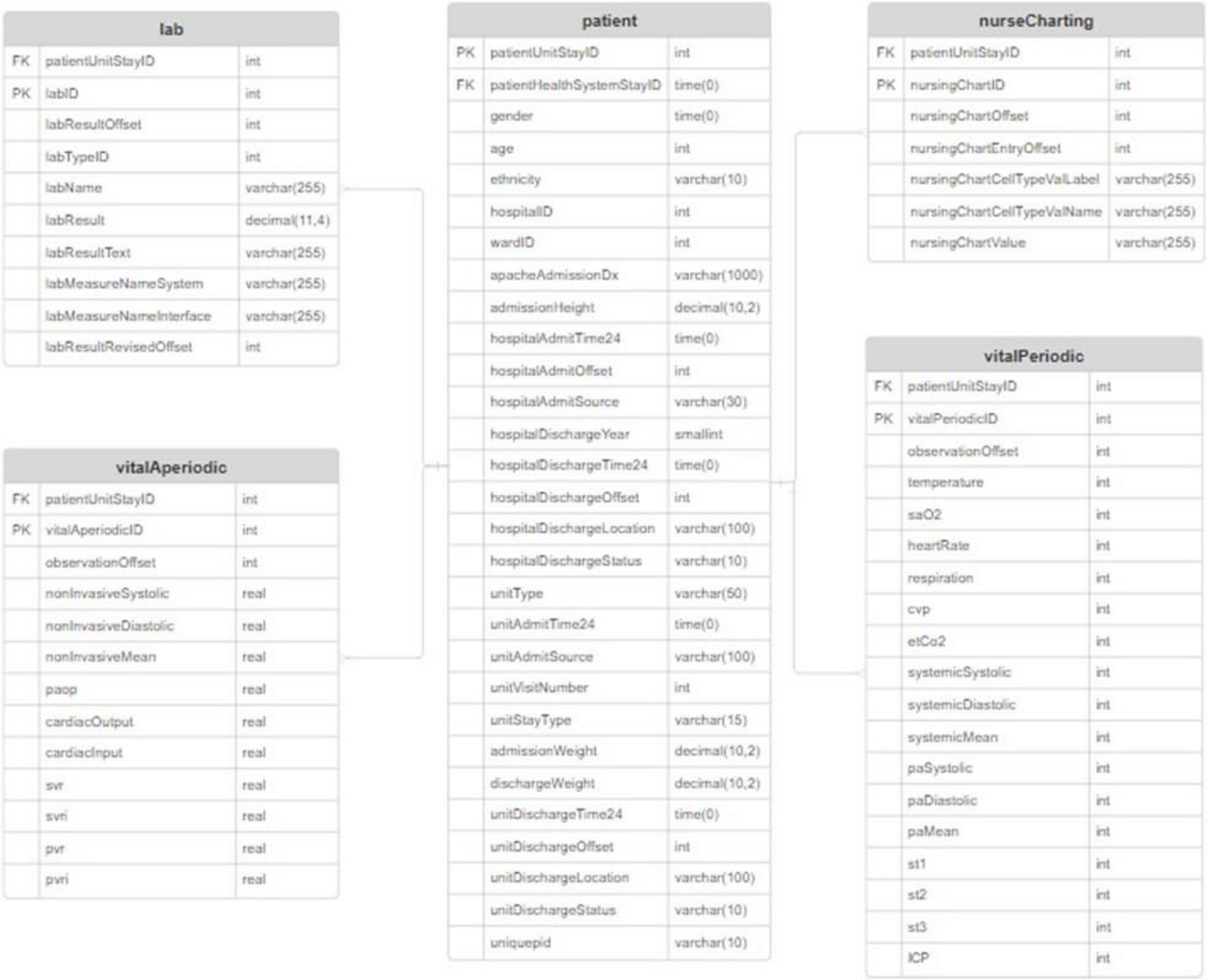
Database description: https://eicu-crd.mit.edu/about/eicu/

Objectives

We tried to re-create the findings from the article below by using machine learning to build predictive model to predict the status of partient in first admission to the ICU:

- Peer review article as reference: Na Pattalung, T., Ingviya, T., & Chaichulee, S. (2021). Feature explanations in recurrent neural networks for predicting risk of mortality in intensive care patients. Journal of Personalized Medicine, 11(9), 934.

ERD for SQL query



lab

FK	patientUnitStayID	int
PK	labID	int
	labResultOffset	int
	labTypeID	int
	labName	varchar(255)
	labResult	decimal(11,4)
	labResultText	varchar(255)
	labMeasureNameSystem	varchar(255)
	labMeasureNameInterface	varchar(255)
	labResultRevisedOffset	int

vitalAperiodic

FK	patientUnitStayID	int
PK	vitalAperiodicID	int
	observationOffset	int
	nonInvasiveSystolic	real
	nonInvasiveDiastolic	real
	nonInvasiveMean	real
	paop	real
	cardiacOutput	real
	cardiacInput	real
	svr	real
	svri	real
	pvr	real
	pvri	real

patient

PK	patientUnitStayID	int
FK	patientHealthSystemStayID	time(0)
	gender	time(0)
	age	int
	ethnicity	varchar(10)
	hospitalID	int
	wardID	int
	apacheAdmissionDx	varchar(1000)
	admissionHeight	decimal(10,2)
	hospitalAdmitTime24	time(0)
	hospitalAdmitOffset	int
	hospitalAdmitSource	varchar(30)
	hospitalDischargeYear	smallint
	hospitalDischargeTime24	time(0)
	hospitalDischargeOffset	int
	hospitalDischargeLocation	varchar(100)
	hospitalDischargeStatus	varchar(10)
	unitType	varchar(50)
	unitAdmitTime24	time(0)
	unitAdmitSource	varchar(100)
	unitVisitNumber	int
	unitStayType	varchar(15)
	admissionWeight	decimal(10,2)
	dischargeWeight	decimal(10,2)
	unitDischargeTime24	time(0)
	unitDischargeOffset	int
	unitDischargeLocation	varchar(100)
	unitDischargeStatus	varchar(10)
	uniquepid	varchar(10)

nurseCharting

FK	patientUnitStayID	int
PK	nursingChartID	int
	nursingChartOffset	int
	nursingChartEntryOffset	int
	nursingChartCellTypeValLabel	varchar(255)
	nursingChartCellTypeValName	varchar(255)
	nursingChartValue	varchar(255)

vitalPeriodic

FK	patientUnitStayID	int
PK	vitalPeriodicID	int
	observationOffset	int
	temperature	int
	saO2	int
	heartRate	int
	respiration	int
	cvp	int
	etCo2	int
	systemicSystolic	int
	systemicDiastolic	int
	systemicMean	int
	paSystolic	int
	paDiastolic	int
	paMean	int
	st1	int
	st2	int
	st3	int
	ICP	int

Based on the reference paper, main features were collected mostly from 3 tables: PATIENT, LAB and VITALPERIODIC

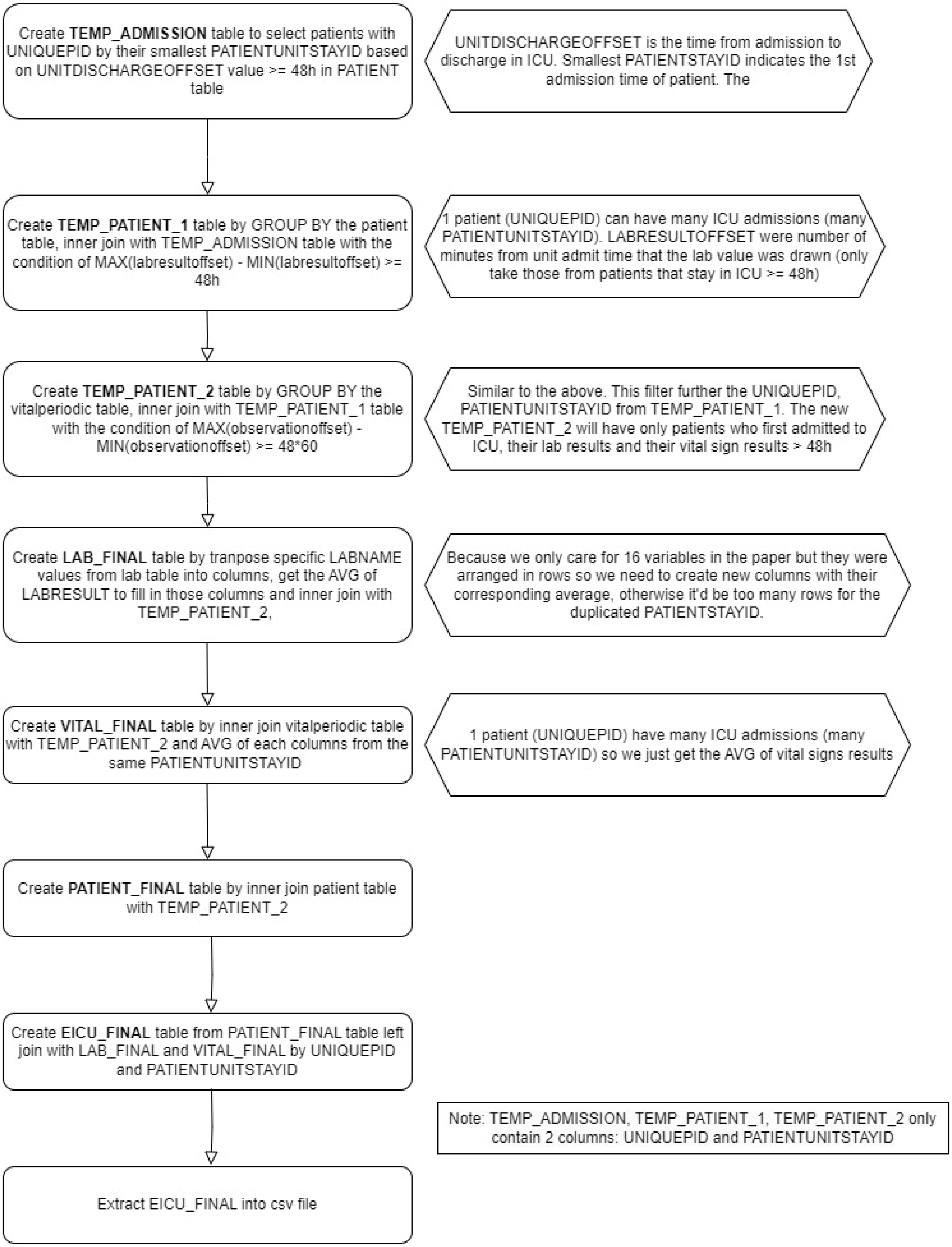
Other classmates used NURSECHARTING and VITALAPERIODC tables to fill in missing values for test results such as temperature but blindly use them without understanding health data is risky so we only chose 3 main tables for query. For instance, in VITALAPERIODIC table, they had invasive and non-invasive DIASTOLIC results but in VITALPERIODIC table, they wrote Systemic Diastolic so we were unsure which can be used. On another hands, NURSECHARTING only provided a few lab results but they were all in time series (depends on offset timestamps) so there are higher chance of getting wrong results to fill in missing values.

Tables description for query

Table name	Description	Time Series
patient	Contains comprehensive demographic data (age, gender, ethnicity) and details regarding hospital admissions and discharges, such as dates, unit types, and diagnoses. This information is crucial for understanding patient backgrounds, managing hospital resources, and conducting epidemiological research.	Yes, tracks patient admission and discharge events with hospitalAdmitOffset and hospitalDischargeOffset. There is also ICU discharge events with unitDischargeOffset (number of minutes from unit admit time that the patient was discharged from the unit), all these columns are having the values in minutes. -hospitalAdmitTime24: This captures the time of the patient's hospital admission in a 24-hour format (e.g., "12:45"). -hospitalDischargeTime24: This captures the time of the patient's hospital discharge, also in a 24-hour format. - unitAdmitTime24: Records the time the patient was admitted to a specific unit within the hospital, again using a 24-hour format. - unitDischargeTime24: This captures the time of the patient's discharge from the specific unit in a 24-hour format.
lab	Laboratory tests that have been mapped to a standard set of measurements (albumin, bilirubin, lactate, etc.). Unmapped measurements are recorded in the customLab table. This table has 10 features and 39.1 mil observations. It links with "patient" table by FK of PATIENTUNITSTAYID which related to the time series data in "patient". This is where 16 features of lab from the paper	This table is time series data because it has labResultOffset where they received test result at different time Duplicated PATIENUNITSTAYID since there were many lab tests per patient per ICU stay. For each lab result, add the labResultOffset (number of minutes from unit admit time) to the unitAdmitTime24 value from the "PATIENT" table.

	was taken from, filtering by “labName” column	
vitalPeriodic	“Periodic” data refers to data which is consistently interfaced from bedside vital signs monitors into eCareManager. Vital signs are referred to as periodic vital signs as they are captured continuously by the system. 19 features with vital signs as observationOffset, temperature, saO2, heartRate, respiration, cvp, etCo2, systemicSystolic, systemicDiastolic, systemicMean, paSystolic, paDiastolic, paMean, st1, st2, st3, ICP vitalPeriodicID is the PK of this table patientUnitStayID is the FK of this table which link with the Patient table	Yes, Data are generally interfaced as 1 minute averages, and archived into the vitalPeriodic table as 5 minute median values. vitalPeriodic data represents the 5 minute median values from the bedside vital signs monitors, and is therefore unvalidated data. Unvalidated implies that the data has not been checked and verified by a bedside care provider, i.e. the measurements may be noisy and not reflect true patient state.

Flowchart



Create **TEMP_ADMISSION** table to select patients with UNIQUEPID by their smallest PATIENTUNITSTAYID based on UNITDISCHARGEOFFSET value >= 48h in PATIENT table

UNITDISCHARGEOFFSET is the time from admission to discharge in ICU. Smallest PATIENTSTAYID indicates the 1st admission time of patient. The

Create **TEMP_PATIENT_1** table by GROUP BY the patient table, inner join with TEMP_ADMISSION table with the condition of MAX(labresultoffset) - MIN(labresultoffset) >= 48h

1 patient (UNIQUEPID) can have many ICU admissions (many PATIENTUNITSTAYID). LABRESULTOFFSET were number of minutes from unit admit time that the lab value was drawn (only take those from patients that stay in ICU >= 48h)

Create **TEMP_PATIENT_2** table by GROUP BY the vitalperiodic table, inner join with TEMP_PATIENT_1 table with the condition of MAX(observationoffset) - MIN(observationoffset) >= 48*60

Similar to the above. This filter further the UNIQUEPID, PATIENTUNITSTAYID from TEMP_PATIENT_1. The new TEMP_PATIENT_2 will have only patients who first admitted to ICU, their lab results and their vital sign results > 48h

Create **LAB_FINAL** table by tranpose specific LABNAME values from lab table into columns, get the AVG of LABRESULT to fill in those columns and inner join with TEMP_PATIENT_2,

Because we only care for 16 variables in the paper but they were arranged in rows so we need to create new columns with their corresponding average, otherwise it'd be too many rows for the duplicated PATIENTSTAYID.

Create **VITAL_FINAL** table by inner join vitalperiodic table with TEMP_PATIENT_2 and AVG of each columns from the same PATIENTUNITSTAYID

1 patient (UNIQUEPID) have many ICU admissions (many PATIENTUNITSTAYID) so we just get the AVG of vital signs results

Create **PATIENT_FINAL** table by inner join patient table with TEMP_PATIENT_2

Create **EICU_FINAL** table from PATIENT_FINAL table left join with LAB_FINAL and VITAL_FINAL by UNIQUEPID and PATIENTUNITSTAYID

Extract EICU_FINAL into csv file

Note: TEMP_ADMISSION, TEMP_PATIENT_1, TEMP_PATIENT_2 only contain 2 columns: UNIQUEPID and PATIENTUNITSTAYID

SQL codes

```
------------------------------------------
--Create temp_admission table to select patient with uniquepid with their SMALLEST patientunitstayID based on their unitdischareoffset >= 48h
------------------------------------------
create table if exists temp_admission
AS
WITH first_admission AS (
  SELECT
    uniquepid,
    patientUnitStayID,
               unitdischargeoffset,
    ROW_NUMBER() OVER (PARTITION BY uniquepid ORDER BY patientUnitStayID) AS rn
  FROM patient)
SELECT
  uniquepid,
  patientUnitStayID
FROM first_admission
WHERE rn = 1
AND unitdischargeoffset >= 48 * 60;
------------------------------------------
--Select lab table with conditions of having same uniquepid in temp_patients and max - min >= 48h
------------------------------------------
create or replace temporary table temp_patient_1 as
SELECT
  l.patientunitstayID,
        p.uniquepid,
  MAX(labresultoffset) AS max_offset,
  MIN(labresultoffset) as min_offset
FROM
  lab l
JOIN temp_admission p
on l.patientunitstayID = p.patientunitstayID
GROUP BY
  l.patientunitstayID, p.uniquepid
having
  MAX(labresultoffset) - MIN(labresultoffset) >= 48*60;
---------------------------------------------
```

---Select vitalperiodic table with conditions of having same uniquepid in temp_patients and max - min >= 48h

```
-------------------------------------------
create or replace temporary table temp_patient_2 as
SELECT
  v.patientunitstayID,
        tp1.uniquepid,
  MAX(observationoffset) AS max_offset,
  MIN(observationoffset) as min_offset
FROM
  vitalperiodic v
JOIN temp_patient_1 tp1
on v.patientunitstayID = tp1.patientunitstayID
GROUP BY
  v.patientunitstayID, tp1.uniquepid
```

```
having

  MAX(observationoffset) - MIN(observationoffset) >= 48*60;

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

--Create final lab table

--Because we only care for 17 variables but they arranged it in rows so we need to create new columns with their corresponding average, otherwise it'd be too many rows for a single patienunitstayid. We also considered using MEDIAN instead of AVG to avoid outliers but since the data was grouped by from a single patient (uniquepid) so we kept it.

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

create or replace temporary table lab_final as

SELECT uniquepid, patientunitstayid,

    AVG(CASE WHEN labname = 'albumin' THEN labresult_avg END) AS Albumin,

    AVG(CASE WHEN labname = 'BUN' THEN labresult_avg END) AS Blood_urea_nitrogen,

    AVG(CASE WHEN labname = 'total bilirubin' THEN labresult_avg END) AS Bilirubin,

    AVG(CASE WHEN labname = 'lactate' THEN labresult_avg END) AS Lactate,

    AVG(CASE WHEN labname = 'bicarbonate' THEN labresult_avg END) AS Bicarbonate,

    AVG(CASE WHEN labname = '-bands' THEN labresult_avg END) AS Band_neutrophil,

    AVG(CASE WHEN labname = 'chloride' THEN labresult_avg END) AS Chloride,

    AVG(CASE WHEN labname = 'creatinine' THEN labresult_avg END) AS Creatinine,

    AVG(CASE WHEN labname = 'glucose' THEN labresult_avg END) AS Glucose,

    AVG(CASE WHEN labname = 'Hgb' THEN labresult_avg END) AS Hemoglobin,

    AVG(CASE WHEN labname = 'Hct' THEN labresult_avg END) AS Hematocrit,

    AVG(CASE WHEN labname = 'platelets x 1000' THEN labresult_avg END) AS
Platelet_count,

    AVG(CASE WHEN labname = 'potassium' THEN labresult_avg END) AS Potassium,

    AVG(CASE WHEN labname = 'PTT' THEN labresult_avg END) AS
Partial_thromboplastin_time,

    AVG(CASE WHEN labname = 'sodium' THEN labresult_avg END) AS Sodium,
    AVG(CASE WHEN labname = 'WBC x 1000' THEN labresult_avg END) AS White_blood_cells

FROM (

  SELECT

  tp2.patientunitstayid, tp2.uniquepid,

  l.labname, AVG(l.labresult) as labresult_avg

FROM

  temp_patient_2 tp2

INNER JOIN

  lab l ON tp2.patientunitstayid = l.patientunitstayid

WHERE l.labname IN ('albumin','BUN','total bilirubin','lactate','glucose','platelets x 1000','Hgb','WBC x 1000','chloride','PTT','creatinine','bicarbonate','potassium','sodium','Hct','-bands')

GROUP BY

  tp2.patientunitstayid, tp2.uniquepid, l.labname)


GROUP BY patientunitstayid, uniquepid;

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

--Noticed that there are only 10.6k rows of band_neutrofil lab results over total 49.9k rows so there are a lot of missing values since uniquepid and patientunitstayid from temp_patient_2 didn't have it

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

SELECT

  tp2.patientunitstayid, tp2.uniquepid,

  l.labname, AVG(l.labresult) as labresult_avg

FROM

  temp_patient_2 tp2

INNER JOIN

  lab l ON tp2.patientunitstayid = l.patientunitstayid
```

```
WHERE l.labname IN ('-bands')

GROUP BY

   tp2.patientunitstayid, tp2.uniquepid, l.labname;

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

--Create final vitalperiodic table by filtering the uniquepid and patientunitstayid from temp_patient_2

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

select * from vitalperiodic;

create or replace temporary table vital_final as

SELECT

   tp2.patientunitstayid, tp2.uniquepid,

  AVG(vp.temperature) as temperature, AVG(vp.sao2) as saturate_oxygen,
AVG(vp.heartrate) as heartrate, AVG(vp.respiration) as respiration,
AVG(vp.systemicdiastolic) as diastolic, AVG(vp.systemicmean) as mean_arterial,
AVG(vp.systemicsystolic) as systolic

FROM

   temp_patient_2 tp2

INNER JOIN

   vitalperiodic vp ON tp2.patientunitstayid = vp.patientunitstayid

group by

   tp2.patientunitstayid, tp2.uniquepid;


select * from vital_final; --See the result


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

--Create final patient table by filtering the uniquepid and patientunitstayid from temp_patient_2

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

create or replace temporary table patient_final as
```

```
SELECT

   tp2.patientunitstayid, tp2.uniquepid,

   p.gender, p.age, p.ethnicity, p.admissionheight, p.unittype, p.admissionweight,
p.dischargeweight, p.unitdischargeoffset, p.unitdischargestatus,

FROM

   temp_patient_2 tp2

INNER JOIN

   patient p ON tp2.uniquepid = p.uniquepid and tp2.patientunitstayid = p.patientunitstayid

ORDER BY

   tp2.uniquepid;

--------------------------------------------

--Join all 3 final tables to create the final table for EICU database

--------------------------------------------


Create or replace temporary table EICU_final as

select

   pf.PATIENTUNITSTAYID, pf.UNIQUEPID, pf.GENDER, pf.AGE, pf.ETHNICITY,
pf.ADMISSIONHEIGHT, pf.UNITTYPE, pf.ADMISSIONWEIGHT, pf.DISCHARGEWEIGHT,
pf.UNITDISCHARGEOFFSET, pf.UNITDISCHARGESTATUS,

   vf.TEMPERATURE, vf.SATURATE_OXYGEN, vf.HEARTRATE, vf.RESPIRATION, vf.DIASTOLIC,
vf.MEAN_ARTERIAL, vf.SYSTOLIC,

   lf.ALBUMIN,           lf.BLOOD_UREA_NITROGEN, lf.BILIRUBIN, lf.LACTATE,
lf.BICARBONATE, lf.BAND_NEUTROPHIL, lf.CHLORIDE,  lf.CREATININE, lf.GLUCOSE,
          lf.HEMOGLOBIN, lf.HEMATOCRIT, lf.PLATELET_COUNT, lf.POTASSIUM,
lf.PARTIAL_THROMBOPLASTIN_TIME, lf.SODIUM, lf.WHITE_BLOOD_CELLS

from

   patient_final pf

left join

   lab_final lf on pf.uniquepid = lf.uniquepid and pf.patientunitstayid = lf.patientunitstayid
```

```
left join
   vital_final vf on pf.uniquepid = vf.uniquepid and pf.patientunitstayid = vf.patientunitstayid
order by
   pf.uniquepid;
-------------------------------------------
--See the result
-------------------------------------------
select * from EICU_final;
```