Part 1

F1 = {ABC ⟶ D, BC ⟶ EA, BCE ⟶ G }
F2 = {A⟶B, C⟶AD, AE⟶CG, BC ⟶ C}
F3 = {AC ⟶ B, BC ⟶ D, BD ⟶ E, AE ⟶ G, ED ⟶ A, DA ⟶ C}

For BCNF

F1:

ABC -> D {ABCD}
BC -> EA {BCEA}
BCE -> G {BCEG}

for ABC
now, BC -> EA, so ABC -> ABCDE
BCE -> G, so ABC -> ABCDEG

for BC:
BCE -> BCEG
ABC -> ABCD
hence, BCE -> ABCDEG

for BCE:
since BC -> ABCDEG
BCE -> ABCDEG

all three in F1, left hand side is superkey hence F1 is in BCNF

F2:

A -> B
C -> AD
AE -> CG
BC ⟶ C

A -> B only hence F2 is not in BCNF

F3: {AC ⟶ B, BC ⟶ D, BD ⟶ E, AE ⟶ G, ED ⟶ A, DA ⟶ C}

AC -> ABC
BC -> BCD
BD -> BDE

AE -> AEG
ED -> EAD
DA -> DAC

for AC,

AC -> ABC
BC -> BCD, so AC -> ABCD
BD -> BDE, so AC -> ABCDE
AE -> AEG, so AC -> ABCDEG hence AC is superkey

for BC,

BC -> BCD
BD -> BDE, so BC -> BCDE
ED -> EAD, so BC -> BCEAD
AE -> AEG, so BC -> ABCDEG

AE is superkey

for

BD -> BDE
DE -> EAD, so BD -> BEAD
AE -> AEG, so BD -> ABEDG
DA -> DAC, so BD -> ABCDEG

for

AE -> AEG

since it can't derive all tuples, F3 is not in BCNF

For Candidate key:

AC → B, BC → D, BD → E, AE → G, ED → A, DA → C

F1: as derived earlier, BC is superkey, also it can't be decomposed further, No key is smaller than this which is superkey

hence BC is candidate key

F2:

A -> B
C -> AD
AE -> CG
BC → C

AE -> CG {AECG}
C -> AD, so AE -> ACDEG
A -> B so, AE -> ABCDEG

AE is candidate key

CE -> CEADBG

CE is candidate key

- AE, CE are candidate keys

F3:

AC → B, BC → D, BD → E, AE → G, ED → A, DA → C

as proved earlier, only AE is not superkey, rest all are superkeys.

For Lossless decomposition -

Consider a relation schema R(A, B, C, D, E, G) and the following sets of functional dependencies
F1 = {ABC → D, BC → EA, BCE → G }
F2 = {A→B, C→AD, AE→CG, BC → C}, AE, CE
F3 = {AC → B, BC → D, BD → E, AE → G, ED → A, DA → C},

Consider partitioning R into 3 sub relations R1{A,B,C} , R2{D,E,G} , R3{B,C,D}. Is this

decomposition lossless? Explain your answer.

Let's consider R2 and R3,

R2 U R3 -> gives B,C,D,E,G
R2 intersection R3 gives D

for any relation D doesn't derive R2 - R3 (E,G) or R3 - R2 (B,C)
hence R2, R3 is itself not lossless

Hence R1, R2, R3 will not be lossless

Part 2

Relations:
person (person_id,name,dob,gender)
employee(person_id, schedule,employee_type,salary per hour)
member(person_id, membership_id)

Not Null Attributes:
All non-key attributes are not NULL
.
Inclusion Constraints:
employee(person_id) person(person_id)
member(person_id) is subset of person(person_id)

Relations:
trainer(person_id, credentials)
desk_employee(person_id) -> attributes are not repeated from parent table as we have created separate parent table

Not Null Attributes:
All non-key attributes are not NULL

Inclusion Constraints:
trainer(person_id) employee(person_id)
desk_employee(person_id) employee(person_id)

Relations:
university _affiliate(person_id, membership_id,department)
family(person_id, membership_id,credit_card),
related(person_id_uni, person_id_family)

Not Null Attributes:
All non-key attributes are not NULL

Inclusion constraints:
related(person_id_uni) university _affiliate (person_id)
related(person_id_family) family (person_id)
family (person_id) related(person_id_family)
univeristy_affiliate(person_id) member(person_id)
family(person_id) member(person_id)

Relations:
non-student(person_id, member_type,credit_card)
student(person_id, student_type)

Not Null Attributes:
All non-key attributes are not NULL

Inclusion constraints:
non-student(person_id) university _affiliate(person_id)
student(person_id) university _affiliate(person_id)

*Relation*s:
employee_exit_log(person_id,timestamp)
entry_log(person_id, timestamp)

Not Null Attributes:
All non-key attributes are not NULL

Inclusion constraints:
employee_exit_log(person_id) person(person_id)
entry_log(person_id) is subset of person(person_id)

Relations:
events(event_id,description,start_time,end_time,capacity)
space(space_id, description,max_capacity)
equipment(equipment_id , equipment_type,is_available)

hosted_in(event_id, space_id)
contains(space_id, equipment_id)
attends(membership_id, event_id)

Not Null Attributes:
All non-key attributes are not NULL

Inclusion constraints:
hosted_in(event_id) events(event_id)
hosted_in(space_id) space(space_id)

events(event_id) hosted_in(event_id)

contains(space_id) space(space_id)
contains(equipment_id) equipment(equipment_id)
equipment(equipment_id) contains(equipment_id)

attends(membership_id) member(membership_id)
attends(event_id) events(event_id)

Relations:

location sensor(sensor_id,coverage)
equipment sensor(sensor_id,coverage)

usage_reading(equipment_id, member_id, equipment_sensor_id, timestamp)
location_reading(space_id, person_id, location_sensor_id, timestamp)

Not Null Attributes:
All non-key attributes are not NULL

Inclusion constraints:
usage_reading(equipment_id) equipment(equipment_id)
usage_reading(membership_id) member(membership_id)
usage_reading(equipment_sensor_id) equipment_sensor(equipment_sensor_id)
location_reading(space_id) space(space_id)
location_reading(person_id) person(person_id)
location_reading(location_sensor_id) locationt_sensor(location_sensor_id)

Queries -

```
CREATE TABLE Person
(person_id INTEGER NOT NULL,
name VARCHAR(40) NOT NULL,
dob DATE NOT NULL,
gender VARCHAR(40) NOT NULL,
PRIMARY KEY (person_id));


CREATE TABLE Employee(person_id INTEGER NOT NULL,
employee_schedule varchar(40),
employee_type ENUM("Trainer", "desk_employee") NOT NULL, -- not sure
salary_per_hour DECIMAL(2,2) NOT NULL,
PRIMARY KEY (person_id),
FOREIGN KEY (person_id) REFERENCES Person(person_id));


CREATE TABLE Member_main(person_id INTEGER NOT NULL, -- member_main is written
because member is keyword in sql
membership_id INTEGER NOT NULL,
PRIMARY KEY (person_id),  -- should we keep membership id also as primary key?
FOREIGN KEY (person_id) REFERENCES Person(person_id)
);

CREATE TABLE university_affiliate(person_id INTEGER NOT NULL, -- member_main is written
because member is keyword in sql
membership_id INTEGER NOT NULL,
department varchar(40),  -- should we keep membership id also as primary key?
PRIMARY KEY (person_id),
FOREIGN KEY (person_id) REFERENCES Member_main(person_id)
);

CREATE TABLE family(person_id INTEGER NOT NULL, -- member_main is written because
member is keyword in sql
membership_id INTEGER NOT NULL,
credit_card_info varchar(40),
PRIMARY KEY (person_id),
FOREIGN KEY (person_id) REFERENCES Member_main(person_id)
);

CREATE TABLE trainer(person_id INTEGER NOT NULL, -- member_main is written because
member is keyword in sql
```

```
credentials varchar(40),
PRIMARY KEY (person_id),
FOREIGN KEY (person_id) REFERENCES Employee(person_id)
);

CREATE TABLE desk_employee(person_id INTEGER NOT NULL, -- member_main is written
because member is keyword in sql
PRIMARY KEY (person_id),
FOREIGN KEY (person_id) REFERENCES Employee(person_id)
);

CREATE TABLE student(person_id INTEGER NOT NULL, -- member_main is written because
member is keyword in sql
student_type varchar(40),
PRIMARY KEY (person_id),
FOREIGN KEY (person_id) REFERENCES Member_main(person_id)
);

CREATE TABLE non_student(person_id INTEGER NOT NULL, -- member_main is written
because member is keyword in sql
department varchar(40),  -- should we keep membership id also as primary key?
credit_card varchar(40),
PRIMARY KEY (person_id),
FOREIGN KEY (person_id) REFERENCES Member_main(person_id)
);

CREATE TABLE related(person_id_university INTEGER NOT NULL,
person_id_family INTEGER NOT NULL,
PRIMARY KEY (person_id_family),
FOREIGN KEY (person_id_family) REFERENCES family(person_id),
FOREIGN KEY (person_id_university) REFERENCES University_affiliate(person_id));

CREATE TABLE Entry_log(person_id INTEGER NOT NULL,
 entry_timestamp datetime NOT NULL,
PRIMARY KEY (person_id, entry_timestamp),  -- to make timestamp also as a part of primary
key?
FOREIGN KEY (person_id) REFERENCES Person(person_id));


CREATE TABLE Exit_log(employee_id INTEGER NOT NULL,
 exit_timestamp datetime NOT NULL,
PRIMARY KEY (employee_id, exit_timestamp),
FOREIGN KEY (employee_id) REFERENCES Employee(person_id));
```

```
CREATE TABLE Space(space_ID INTEGER NOT NULL,
 space_description VARCHAR(100) NOT NULL,
 Max_Capacity INTEGER NOT NULL,
 PRIMARY KEY (space_id));

CREATE TABLE Events_conducted(event_ID INTEGER NOT NULL,
event_description VARCHAR(100) NOT NULL,
start_time TIME NOT NULL,
end_time TIME NOT NULL,
capacity INTEGER NOT NULL,
PRIMARY KEY (event_ID));

CREATE TABLE Equipment(equipment_ID INTEGER NOT NULL,
equipment_type VARCHAR(40) NOT NULL,
is_available BIT NOT NULL,
PRIMARY KEY (equipment_id));


CREATE TABLE Location_sensor(sensor_id INTEGER NOT NULL, coverage VARCHAR(40)
NOT NULL,
PRIMARY KEY (sensor_id));

CREATE TABLE Equipment_sensor(sensor_id INTEGER NOT NULL, coverage VARCHAR(40)
NOT NULL,
PRIMARY KEY (sensor_id));

CREATE TABLE Hosted_in(event_id INTEGER NOT NULL, space_id INTEGER NOT NULL,
PRIMARY KEY (event_id),
FOREIGN KEY (event_id) REFERENCES Events_conducted(event_id),
FOREIGN KEY (space_id) REFERENCES Space(space_id));

CREATE TABLE space_cotains(space_id INTEGER NOT NULL, equipment_id INTEGER NOT
NULL,
PRIMARY KEY (equipment_id),
FOREIGN KEY (equipment_id) REFERENCES Equipment(equipment_id),
FOREIGN KEY (space_id) REFERENCES Space(space_id));

CREATE TABLE Attends(event_id INTEGER NOT NULL, member_id INTEGER NOT NULL,
PRIMARY KEY (member_id),
FOREIGN KEY (event_id) REFERENCES Events_conducted(event_id),
FOREIGN KEY (member_id) REFERENCES Member_main(person_id)); -- membership_id or
person_id?

CREATE TABLE Usage_Reading(equipment_id INTEGER NOT NULL,
```

```
 member_id INTEGER NOT NULL,
 sensor_id INTEGER NOT NULL,
 usage_timestamp timestamp,
 PRIMARY KEY(equipment_id, member_id, sensor_id, usage_timestamp),
FOREIGN KEY (sensor_id) REFERENCES Equipment_sensor(sensor_id),
FOREIGN KEY (member_id) REFERENCES Member_main(person_id),
FOREIGN KEY (equipment_id) REFERENCES Equipment(equipment_id));

CREATE TABLE Location_Reading(equipment_id INTEGER NOT NULL,
 member_id INTEGER NOT NULL,
 sensor_id INTEGER NOT NULL,
 location_timestamp timestamp,
 PRIMARY KEY(equipment_id, member_id, sensor_id, location_timestamp),
FOREIGN KEY (sensor_id) REFERENCES Location_sensor(sensor_id),
FOREIGN KEY (member_id) REFERENCES Member_main(person_id),
FOREIGN KEY (equipment_id) REFERENCES Equipment(equipment_id));
```