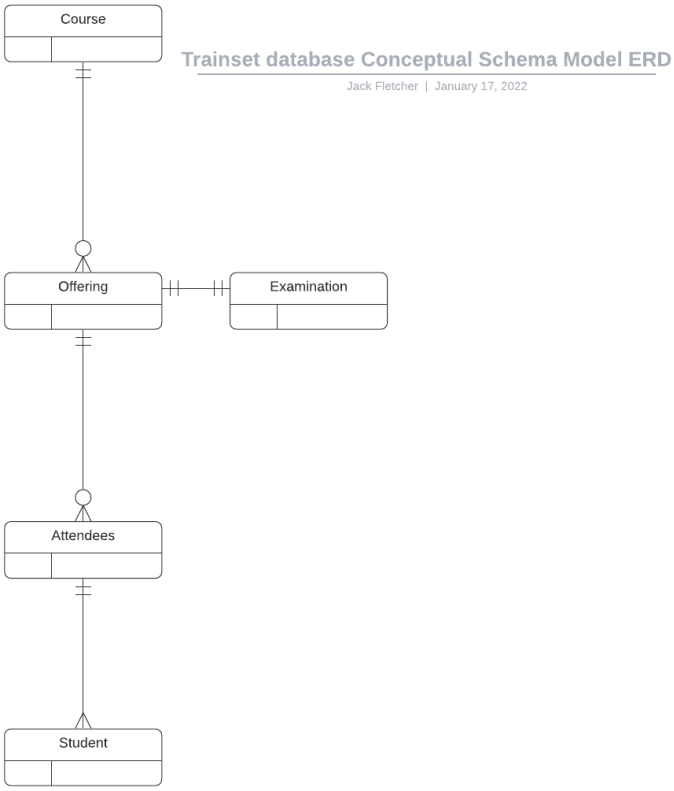
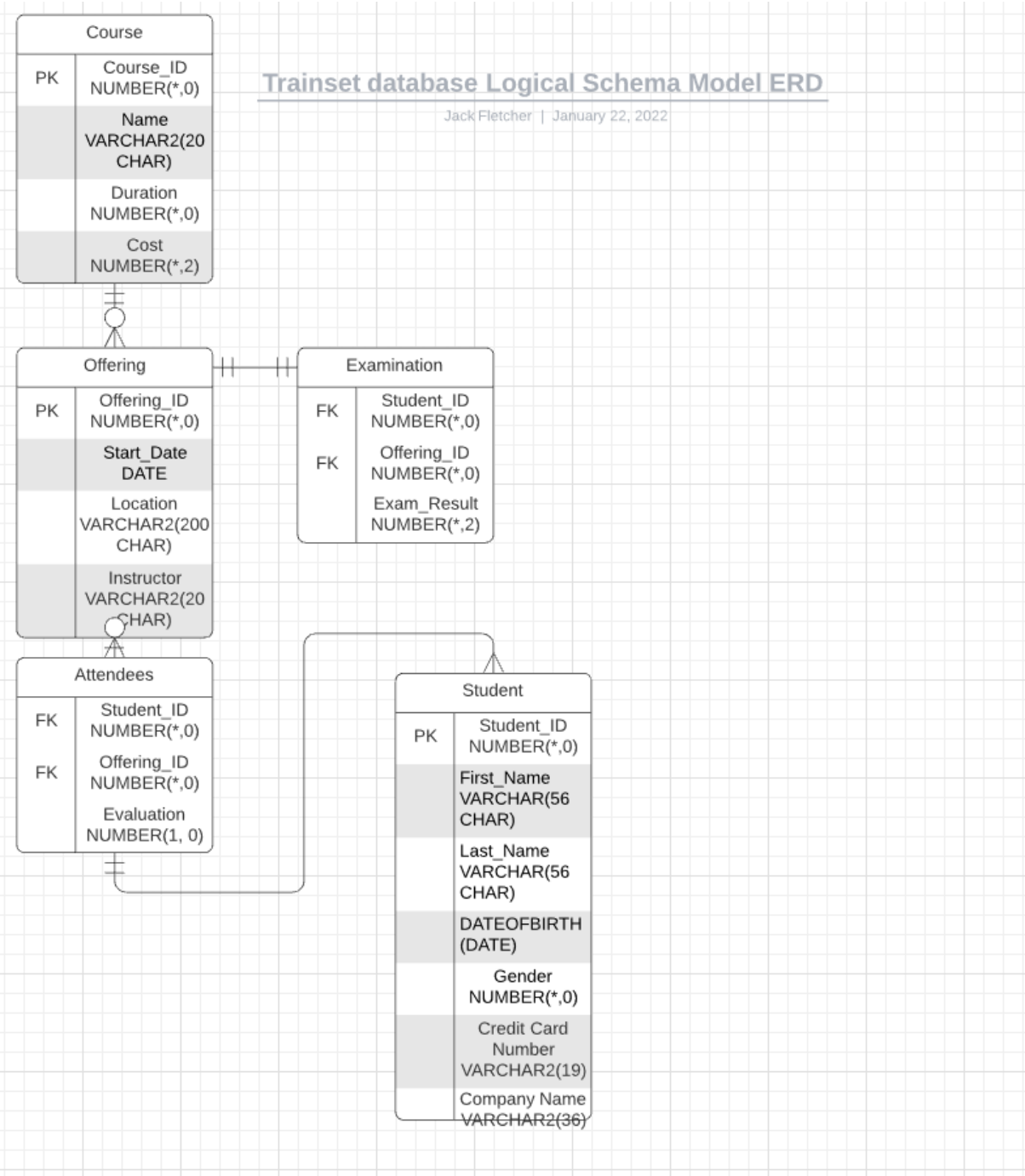
Database Administration and Security

Question 1.






```
-- Init tables
-- Naming conventions:
-- Foreign Key: FK_ForeignKeyTable_PrimaryKeyTable
-- Primary Key: PK_PrimaryKeyTable
-- Constraint: CHECK Name_CHK
ALTER SESSION SET NLS_DATE_FORMAT = "DD-MM-YYYY";
```

```
CREATE TABLE Course(Course_ID NUMBER(*,0),
    Course_Name VARCHAR(200) NOT NULL UNIQUE,
    Course_Duration NUMBER(*,0) NOT NULL,
    Course_Cost NUMBER(*,2) NOT NULL,
    CONSTRAINT PK_Course PRIMARY KEY(Course_ID),
    CONSTRAINT Course_Duration_CHK CHECK (Course_Duration> 0 AND
Course_Duration <= 5),
    CONSTRAINT Course_Cost_CHK CHECK (Course_Cost>= 300.00 AND
Course_Cost <= 2000.00)
);


CREATE TABLE Offering(Offering_ID NUMBER(*,0),
    Course_ID NUMBER(*,0) NOT NULL,
    StartDate DATE NOT NULL,
    Offering_Location VARCHAR2(200 CHAR) NOT NULL,
    Instructor VARCHAR2(80 CHAR) NOT NULL,

    CONSTRAINT PK_Offering PRIMARY KEY(Offering_ID),
    CONSTRAINT FK_Offering_Course FOREIGN KEY(Course_ID) REFERENCES
Course(Course_ID)
);


CREATE TABLE Student(Student_ID NUMBER(*,0),
    First_Name VARCHAR2(56 CHAR) NOT NULL,
    Last_Name VARCHAR2(56 CHAR) NOT NULL,
    DateOfBirth DATE NOT NULL,
    -- Derived from iso standard
https://www.iso.org/standard/36266.html
    -- 0 = Not known,
    -- 1 = Male,
    -- 2 = Female,
    -- 3 = N/A
    Gender NUMBER(*,0) NOT NULL,
    Credit_Card_Number VARCHAR2(19 CHAR) NOT NULL,
```

```
    Company_Name VARCHAR2(36 CHAR),


    CONSTRAINT PK_Student PRIMARY KEY(Student_ID),

    CONSTRAINT Gender_CHK CHECK (Gender >= 0 AND Gender <= 3),

    -- Checks cc number is correct format -> #### #### #### #### */

    CONSTRAINT Credit_Card_Number_CHK CHECK
(REGEXP_LIKE(Credit_Card_Number, '\d{4}\s\d{4}\s\d{4}\s\d{4}'))

);


CREATE TABLE Examination(Student_ID NUMBER(*,0) NOT NULL,

    Offering_ID NUMBER(*,0) NOT NULL,

    Exam_Result NUMBER(*,2),


    CONSTRAINT FK_Examination_Student FOREIGN KEY(Student_ID)
REFERENCES Student(Student_ID),

    CONSTRAINT FK_Examination_Offering FOREIGN KEY(Offering_ID)
REFERENCES Offering(Offering_ID)

);



CREATE TABLE Attendees(Student_ID NUMBER(*,0),

    Offering_ID NUMBER(*,0),

    Student_Evaluation NUMBER(1,0),


    CONSTRAINT FK_Attendees_Offering FOREIGN KEY(Offering_ID)
REFERENCES Offering(Offering_ID),

    CONSTRAINT FK_Attendees_Student FOREIGN KEY(Student_ID)
REFERENCES Student(Student_ID),

    CONSTRAINT Student_Evaluation_CHK CHECK (Student_Evaluation >=0
AND Student_Evaluation <=5)

);
```

Question 2.

```
INSERT INTO COURSE VALUES(1,'Leadership','1','300.00');

INSERT INTO COURSE VALUES(2,'Project Management','2','500.00');

INSERT INTO COURSE VALUES(3,'Comptia','3','700.00');
```

```
INSERT INTO COURSE VALUES(4,'AWS','4','900.00');

INSERT INTO COURSE VALUES(5,'Programming
Fundamentals','5','1200.00');

INSERT INTO COURSE VALUES(6,'Web Development','3','1500.00');


INSERT INTO OFFERING VALUES(1,1,'12-NOV-2021','Pawnee','Ron
Swanson');

INSERT INTO OFFERING VALUES(2,1,'19-NOV-2021','Pawnee','April
Ludgate');

INSERT INTO OFFERING VALUES(3,1,'26-NOV-2021','Pawnee','Leslie
Knope');

INSERT INTO OFFERING VALUES(4,1,'02-FEB-2021','Pawnee','Ron
Swanson');

INSERT INTO OFFERING VALUES(5,1,'09-FEB-2021','Pawnee','April
Ludgate');

INSERT INTO OFFERING VALUES(6,1,'16-FEB-2021','Pawnee','Ben Wyatt');



INSERT INTO OFFERING VALUES(7,2,'01-MAR-2021','London','Mark
Corrigan');

INSERT INTO OFFERING VALUES(8,2,'08-MAR-2021','London','Alan
Johnson');

INSERT INTO OFFERING VALUES(9,2,'15-MAR-2021','London','Jeremy
Usborne');


INSERT INTO OFFERING VALUES(10,3,'06-JAN-2021','Scranton','Michael
Scott');

INSERT INTO OFFERING VALUES(11,3,'13-JAN-2021','Scranton','Pam
Beesly');

INSERT INTO OFFERING VALUES(12,3,'19-JAN-2021','Scranton','Bob
Vance');

INSERT INTO OFFERING VALUES(13,3,'26-JAN-2021','Scranton','Creed
Bratton');


INSERT INTO OFFERING VALUES(14,4,'01-JUN-2021','New York','Jake
Peralta');

INSERT INTO OFFERING VALUES(15,4,'08-JUN-2021','New York','Rosa
Diaz');
```

```
INSERT INTO OFFERING VALUES(16,4,'15-JUN-2021','New York','Amy
Santiago');

INSERT INTO OFFERING VALUES(17,4,'22-JUN-2021','New York','Gina
Linetti');


INSERT INTO OFFERING VALUES(18,5,'01-JUL-2021','Albuquerque','Walter
White');




    -- 0 = Not known,

    -- 1 = Male,

    -- 2 = Female,

    -- 3 = N/A

INSERT INTO STUDENT VALUES(1,'Harry','Potter','06-01-1966','1','1234
5678 9123 4567', 'Gringotts');

INSERT INTO STUDENT VALUES(2,'Hermione','Granger','05-05-
1937','2','1234 5678 9123 4567', 'Ollivanders');

INSERT INTO STUDENT VALUES(3,'Ron','Weasley','01-02-1993','1','1234
5678 9123 4567', 'Ministry of Magic');

INSERT INTO STUDENT VALUES(4,'Draco','Malfoy','01-01-1997','2','1234
5678 9123 4567', 'Gringotts');

INSERT INTO STUDENT
(STUDENT_ID,FIRST_NAME,LAST_NAME,DATEOFBIRTH,GENDER,CREDIT_CARD_NUMB
ER) VALUES(5,'Tom','Riddle','23-12-1987','0','1234 5678 9123 4567');

INSERT INTO STUDENT
(STUDENT_ID,FIRST_NAME,LAST_NAME,DATEOFBIRTH,GENDER,CREDIT_CARD_NUMB
ER) VALUES(6,'Albus','Dumbledore','25-12-1977','1','1234 5678 9123
4567');

INSERT INTO STUDENT
(STUDENT_ID,FIRST_NAME,LAST_NAME,DATEOFBIRTH,GENDER,CREDIT_CARD_NUMB
ER) VALUES(7,'Severus','Snape','12-12-2001','1','1234 5678 9123
4567');

INSERT INTO STUDENT
(STUDENT_ID,FIRST_NAME,LAST_NAME,DATEOFBIRTH,GENDER,CREDIT_CARD_NUMB
ER) VALUES(8,'Minerva','McGonagall','06-06-1966','2','1234 5678 9123
4567');

INSERT INTO STUDENT
(STUDENT_ID,FIRST_NAME,LAST_NAME,DATEOFBIRTH,GENDER,CREDIT_CARD_NUMB
ER) VALUES(9,'Luna','Lovegood','14-12-1983','2','1234 5678 9123
4567');
```

```
INSERT INTO STUDENT
(STUDENT_ID,FIRST_NAME,LAST_NAME,DATEOFBIRTH,GENDER,CREDIT_CARD_NUMB
ER) VALUES(10,'Neville','Longbottom','25-06-1945','1','1234 5678
9123 4567');

INSERT INTO STUDENT
(STUDENT_ID,FIRST_NAME,LAST_NAME,DATEOFBIRTH,GENDER,CREDIT_CARD_NUMB
ER) VALUES(11,'Sirius','Black','09-12-2002','1','1234 5678 9123
4567');


INSERT INTO ATTENDEES VALUES(1,1,3);

INSERT INTO ATTENDEES VALUES(2,2,5);

INSERT INTO ATTENDEES VALUES(3,2,2);


INSERT INTO ATTENDEES VALUES(4,3,3);

INSERT INTO ATTENDEES VALUES(5,4,1);

INSERT INTO ATTENDEES VALUES(6,4,5);

INSERT INTO ATTENDEES VALUES(7,4,0);


INSERT INTO ATTENDEES VALUES(8,5,5);

INSERT INTO ATTENDEES VALUES(9,6,0);

INSERT INTO ATTENDEES VALUES(10,7,1);

INSERT INTO ATTENDEES VALUES(11,8,4);

INSERT INTO EXAMINATION VALUES(1,1,79);

INSERT INTO EXAMINATION VALUES(2,2,40);

INSERT INTO EXAMINATION VALUES(3,2,65);

INSERT INTO EXAMINATION VALUES(4,3,10);

INSERT INTO EXAMINATION VALUES(5,4,90);

INSERT INTO EXAMINATION VALUES(6,4,79);

INSERT INTO EXAMINATION VALUES(7,4,40);

INSERT INTO EXAMINATION VALUES(8,5,65);

INSERT INTO EXAMINATION VALUES(9,6,10);

INSERT INTO EXAMINATION VALUES(10,7,90);

INSERT INTO EXAMINATION VALUES(11,8,90);

--
```

Attendees:

	STUDENT_ID	OFFERING_ID	STUDENT_EVALUATION
1	1	1	3
2	2	2	5
3	3	2	2
4	4	3	3
5	5	4	1
6	6	4	5
7	7	4	0
8	8	5	5
9	9	6	0
10	10	7	1
11	11	8	4

Course:

	COURSE_ID	COURSE_NAME	COURSE_DURATION	COURSE_COST
1	1	Leadership	1	300
2	2	Project Management	2	500
3	3	Comptia	3	700
4	4	AWS	4	900
5	5	Programming Fundamentals	5	1200
6	6	Web Development	3	1500

Offering:

	OFFERING_ID	COURSE_ID	STARTDATE	OFFERING_LOCATION	INSTRUCTOR
1	1	1	12-NOV-21	Pawnee	Ron Swanson
2	2	1	19-NOV-21	Pawnee	April Ludgate
3	3	1	26-NOV-21	Pawnee	Leslie Knope
4	4	1	02-FEB-21	Pawnee	Ron Swanson
5	5	1	09-FEB-21	Pawnee	April Ludgate
6	6	1	16-FEB-21	Pawnee	Ben Wyatt
7	7	2	01-MAR-21	London	Mark Corrigan
8	8	2	08-MAR-21	London	Alan Johnson
9	9	2	15-MAR-21	London	Jeremy Usborne
10	10	3	06-JAN-21	Scranton	Michael Scott
11	11	3	13-JAN-21	Scranton	Pam Beesly
12	12	3	19-JAN-21	Scranton	Bob Vance
13	13	3	26-JAN-21	Scranton	Creed Bratton
14	14	4	01-JUN-21	New York	Jake Peralta
15	15	4	08-JUN-21	New York	Rosa Diaz
16	16	4	15-JUN-21	New York	Amy Santiago
17	17	4	22-JUN-21	New York	Gina Linetti
18	18	5	01-JUL-21	Albuquerque	Walter White

Student:

	STUDENT_ID	FIRST_NAME	LAST_NAME	DATEOFBIRTH	GENDER	CREDIT_CARD_NUMBER	COMPANY_NAME
1	1	Harry	Potter	06-JAN-66	1	1234 5678 9123 4567	Gringotts
2	2	Hermione	Granger	05-MAY-37	2	1234 5678 9123 4567	Ollivanders
3	3	Ron	Weasley	01-FEB-93	1	1234 5678 9123 4567	Ministry of Magic
4	4	Draco	Malfoy	01-JAN-97	2	1234 5678 9123 4567	Gringotts
5	5	Tom	Riddle	23-DEC-87	0	1234 5678 9123 4567	(null)
6	6	Albus	Dumbledore	25-DEC-77	1	1234 5678 9123 4567	(null)
7	7	Severus	Snape	12-DEC-01	1	1234 5678 9123 4567	(null)
8	8	Minerva	McGonagall	06-JUN-66	2	1234 5678 9123 4567	(null)
9	9	Luna	Lovegood	14-DEC-83	2	1234 5678 9123 4567	(null)
10	10	Neville	Longbottom	25-JUN-45	1	1234 5678 9123 4567	(null)
11	11	Sirius	Black	09-DEC-02	1	1234 5678 9123 4567	(null)

Examination:

	STUDENT_ID	OFFERING_ID	EXAM_RESULT
1	1	1	79
2	2	2	40
3	3	2	65
4	4	3	10
5	5	4	90
6	6	4	79
7	7	4	40
8	8	5	65
9	9	6	10
10	10	7	90
11	11	8	90

Question 3.

- Primary key constraints were made on table Course using Course_ID

```
CREATE TABLE Course(Course_ID NUMBER(*,0),
    Course_Name VARCHAR(200) NOT NULL UNIQUE,
    Course_Duration NUMBER(*,0) NOT NULL,
    Course_Cost NUMBER(*,2) NOT NULL,
    CONSTRAINT PK_Course PRIMARY KEY(Course_ID),
    CONSTRAINT Course_Duration_CHK CHECK (Course_Duration> 0 AND Course_Duration <= 5),
    CONSTRAINT Course_Cost_CHK CHECK (Course_Cost>= 300.00 AND Course_Cost <= 2000.00)
);
```

- Primary key constraints were made on table Offering using Offering_ID

```
CREATE TABLE Offering(Offering_ID NUMBER(*,0),
    Course_ID NUMBER(*,0),
    StartDate DATE NOT NULL,
    Offering_Location VARCHAR2(200 CHAR) NOT NULL,
    Instructor VARCHAR2(80 CHAR) NOT NULL,

    CONSTRAINT PK_Offering PRIMARY KEY(Offering_ID),
    CONSTRAINT FK_Offering_Course FOREIGN KEY(Course_ID) REFERENCES Course(Course_ID)
);
```

- Primary key constraints were made on table Student using Student_ID

```
CREATE TABLE Student(Student_ID NUMBER(*,0),
    First_Name VARCHAR2(56 CHAR) NOT NULL,
    Last_Name VARCHAR2(56 CHAR) NOT NULL,
    DateOfBirth DATE NOT NULL,
    -- Derived from iso standard https://www.iso.org/standard/36266.html
    -- 0 = Not known,
    -- 1 = Male,
    -- 2 = Female,
    -- 3 = N/A
    Gender NUMBER(*,0) NOT NULL,
    Credit_Card_Number VARCHAR2(19 CHAR) NOT NULL,
    Company_Name VARCHAR2(36 CHAR),

    CONSTRAINT PK_Student PRIMARY KEY(Student_ID),
    CONSTRAINT Gender_CHK CHECK (Gender >= 0 AND Gender <= 3),
    -- Checks cc number is correct format -> #### #### #### #### */
    CONSTRAINT Credit_Card_Number_CHK CHECK (REGEXP_LIKE(Credit_Card_Number, '\d{4}\s\d{4}\s\d{4}\s\d{4}'))
);
```

- All three primary keys were chosen as those values would be used as criteria for matching other tables and therefore needed to be unique. For example, Course_ID was used on table Course as the primary key as it would be used to relate to offerings through their supplementary Course_ID.
- Foreign key constraints were made on table Offering using Course_ID as it needs to correspond to the primary key in Course.

```
CREATE TABLE Offering(Offering_ID NUMBER(*,0),
    Course_ID NUMBER(*,0),
    StartDate DATE NOT NULL,
    Offering_Location VARCHAR2(200 CHAR) NOT NULL,
    Instructor VARCHAR2(80 CHAR) NOT NULL,

    CONSTRAINT PK_Offering PRIMARY KEY(Offering_ID),
    CONSTRAINT FK_Offering_Course FOREIGN KEY(Course_ID) REFERENCES Course(Course_ID)
);
```

- Foreign key constraints were made on table Examination using Student_ID and Offering_ID as they would be required to access Student and Offering tables.

```
CREATE TABLE Examination(Student_ID NUMBER(*,0),
    Offering_ID NUMBER(*,0),
    Exam_Result NUMBER(*,2),

    CONSTRAINT FK_Examination_Student FOREIGN KEY(Student_ID) REFERENCES Student(Student_ID),
    CONSTRAINT FK_Examination_Offering FOREIGN KEY(Offering_ID) REFERENCES Offering(Offering_ID)
);
```

- Foreign key constraints were made on table Attendees so they could access the table Offering and Student to access student and offering details.

```
CREATE TABLE Attendees(Student_ID NUMBER(*,0),
    Offering_ID NUMBER(*,0),
    Student_Evaluation NUMBER(1,0),

    CONSTRAINT FK_Attendees_Offering FOREIGN KEY(Offering_ID) REFERENCES Offering(Offering_ID),
    CONSTRAINT FK_Attendees_Student FOREIGN KEY(Student_ID) REFERENCES Student(Student_ID),
    CONSTRAINT Student_Evaluation_CHK CHECK (Student_Evaluation >=0 AND Student_Evaluation <=5)
);
```

Question 4.

```
--a.  Find details of all courses running in London //Done

        SELECT * FROM OFFERING WHERE OFFERING_LOCATION = 'London';

--b.  Find the course that runs the greatest number of times. //Done

        SELECT * FROM(SELECT COURSE.COURSE_NAME, OFFERING.COURSE_ID,
COUNT(OFFERING.COURSE_ID) as "Course_Offerings" FROM OFFERING

        INNER JOIN COURSE ON OFFERING.COURSE_ID = COURSE.COURSE_ID

        GROUP BY OFFERING.COURSE_ID, COURSE.COURSE_NAME

        ORDER BY "Course_Offerings" DESC)

        WHERE ROWNUM <= 1;

--c.  Find the total number of attendees for each course. //Done

        --Find the number of times OFFERING_ID comes up in attendees

        --that's the total number of attendees for each offering

        --correlate that to course type

        SELECT COURSE.COURSE_NAME, COUNT(ATTENDEES.OFFERING_ID) FROM
ATTENDEES

            INNER JOIN STUDENT ON ATTENDEES.STUDENT_ID =
STUDENT.STUDENT_ID

            INNER JOIN OFFERING ON ATTENDEES.OFFERING_ID =
OFFERING.OFFERING_ID

            INNER JOIN COURSE ON OFFERING.COURSE_ID =
COURSE.COURSE_ID

            GROUP BY COURSE.COURSE_NAME;

--d.  Show details of the student names and the titles of the
courses that they have attended. //Done

        SELECT CONCAT(CONCAT(STUDENT.FIRST_NAME,'
'),STUDENT.LAST_NAME) as "Student Name", COURSE.COURSE_NAME as
"Course Name"

            FROM ATTENDEES

            INNER JOIN STUDENT ON ATTENDEES.STUDENT_ID =
STUDENT.STUDENT_ID

            INNER JOIN OFFERING ON ATTENDEES.OFFERING_ID =
OFFERING.OFFERING_ID

            INNER JOIN COURSE ON OFFERING.COURSE_ID =
COURSE.COURSE_ID;

--e.  List the title and cost of each course. //Done
```

```
SELECT COURSE_NAME,COURSE_COST FROM COURSE;
```

--f. Add a column to your answer to 4(e) that compares the cost of the course to the average cost

-- i.e. shows the difference between the course cost and the average cost of all courses.

```
SELECT COURSE_NAME, COURSE_COST,(SELECT AVG(COURSE_COST) FROM COURSE) AS AVERAGE, COURSE_COST-(SELECT AVG(COURSE_COST) FROM COURSE) AS DIFFERENCE FROM COURSE group by COURSE_NAME, COURSE_COST;
```

	COURSE_NAME	COURSE_COST	AVERAGE	DIFFERENCE
1	Web Development	1500	850	650
2	Project Management	500	850	-350
3	Programming Fundamentals	1200	850	350
4	Leadership	300	850	-550
5	Comptia	700	850	-150
6	AWS	900	850	50

Question 5.

A.

```
Create VIEW CourseOfferingsLastYear AS
    SELECT COURSE.COURSE_NAME,COURSE.COURSE_ID,COURSE.COURSE_DURATION,COURSE.COURSE_COST,OFFERING.OFFERING_ID,OFFERING.STARTDATE,OFFERING.OFFERING_LOCATION,OFFERING.INSTRUCTOR FROM COURSE
    INNER JOIN OFFERING ON OFFERING.COURSE_ID = COURSE.COURSE_ID
    WHERE OFFERING.STARTDATE >= (SYSDATE - 365)
    ORDER BY OFFERING.STARTDATE;
```

```
Create VIEW CourseOfferingsLastYear AS
    SELECT COURSE.COURSE_NAME,COURSE.COURSE_ID,COURSE.COURSE_DURATION,COURSE.COURSE_COST,OFFERING.OFFERING_ID,OFFERING.STARTDATE,OFFERING.OFFERING_LOCATION,OFFERING.INSTRUCTOR FROM COURSE
    INNER JOIN OFFERING ON OFFERING.COURSE_ID = COURSE.COURSE_ID
    WHERE OFFERING.STARTDATE >= (SYSDATE - 365)
    ORDER BY OFFERING.STARTDATE;

-- VIEW CourseOfferingsLastYear;
```

B.

```
----- UPDATES -------
    --Can't modify Course Duration
```

```
--SQL Error: ORA-01779: cannot modify a column which maps to a
non key-preserved table
UPDATE CourseOfferingsLastYear
    SET COURSE_DURATION = 4
        WHERE COURSE_ID = 3;

--Can't modify Course Duration
--SQL Error: ORA-01779: cannot modify a column which maps to a
non key-preserved table
UPDATE CourseOfferingsLastYear
    SET COURSE_DURATION = 4
        WHERE COURSE_COST = 300;

--Can't modify Course Name
--SQL Error: ORA-01779: cannot modify a column which maps to a
non key-preserved table
UPDATE CourseOfferingsLastYear
    SET COURSE_NAME = 'Testing'
        WHERE COURSE_ID = 3;

--Can't modify Course ID
--SQL Error: ORA-01779: cannot modify a column which maps to a
non key-preserved table
UPDATE CourseOfferingsLastYear
    SET COURSE_ID = 15
        WHERE COURSE_NAME = 'COURSE_NAME';

--Can't modify Offering ID
--ORA-00001: unique constraint (F021280L.PK_OFFERING) violated
UPDATE CourseOfferingsLastYear
    SET OFFERING_ID = 15
        WHERE COURSE_COST = '700';

--Can't modify STARTDATE
```

```
    --SQL Error: ORA-01779: cannot modify a column which maps to a
non key-preserved table
    UPDATE CourseOfferingsLastYear
        SET COURSE_COST = 1500
            WHERE COURSE_ID = 1;

    --Works successfully
    UPDATE CourseOfferingsLastYear
        SET OFFERING_LOCATION = 'Testing'
            WHERE COURSE_ID = 1;
    --Works successfully
    UPDATE CourseOfferingsLastYear
        SET OFFERING_LOCATION = 'Testing NO CRITERIA';

    --Works successfully
    UPDATE CourseOfferingsLastYear
        SET INSTRUCTOR = 'Instructor test'
            WHERE COURSE_ID = 1;

    --Works successfully
    UPDATE CourseOfferingsLastYear
        SET INSTRUCTOR = 'Instructor test NO CRITERIA';

----- DELETES -------
-- Can't delete row as child record found
-- ORA-02292: integrity constraint (F021280L.FK_ATTENDEES_OFFERING)
violated - child record found
DELETE FROM COURSEOFFERINGSLASTYEAR WHERE COURSE_ID = 1;

--Can delete this successfully
--There were no child records for its course ID (There were no
students for this course)
--This also deleted it from OFFERING
DELETE FROM COURSEOFFERINGSLASTYEAR WHERE COURSE_ID = 5;
```

```
--Can delete this successfully

--There were no child records for its course ID (There were no
students for this course)

--This also deleted it from OFFERING

DELETE FROM COURSEOFFERINGSLASTYEAR WHERE COURSE_ID = 4;


--Can delete this successfully

--There were no child records for its course ID (There were no
students for this course)

--This also deleted it from OFFERING

DELETE FROM COURSEOFFERINGSLASTYEAR WHERE COURSE_ID = 3;


----- INSERT -------

--Cannot do this with all values

--SQL Error: ORA-01779: cannot modify a column which maps to a non
key-preserved tabl

INSERT INTO COURSEOFFERINGSLASTYEAR
VALUES('AWS',3,'4','700','10','01-02-21','Scranton','Michael
Scott');


--Cannot do this with just values from COURSE table

--SQL Error: ORA-01779: cannot modify a column which maps to a non
key-preserved table

INSERT INTO
COURSEOFFERINGSLASTYEAR(COURSE_ID,COURSE_NAME,COURSE_DURATION,COURSE
_COST)VALUES(1,'Leadership','1','300.00');


--Cannot do this like this; needs to be different

--SQL Error: ORA-01776: cannot modify more than one base table
through a join view

INSERT INTO
COURSEOFFERINGSLASTYEAR(OFFERING_ID,COURSE_ID,STARTDATE,OFFERING_LOC
ATION,INSTRUCTOR)VALUES(18,'4','03-JUN-21','New York','INSTRUCTOR');


--Success, COURSE_ID is null though so this would need to be done
seperately
```

```
INSERT INTO
COURSEOFFERINGSLASTYEAR(OFFERING_ID,STARTDATE,OFFERING_LOCATION,INST
RUCTOR) VALUES(18,'03-JUN-21','Offering
Location1','TestInstructor');

--part 2 cannot be done, although trying to make a standard inssert
to OFFERING without COURSE_ID

--SQL Error: ORA-01779: cannot modify a column which maps to a non
key-preserved table e.g.

--INSERT INTO OFFERING
(OFFERING_ID,STARTDATE,OFFERING_LOCATION,INSTRUCTOR) VALUES(25,'01-
JUL-2021','Albuquerque','Walter White');

    UPDATE CourseOfferingsLastYear

    SET COURSE_ID = 4 WHERE OFFERING_ID = 18;
```

Ease of reading:

```sql
----- UPDATES -------
    --Can't modify Course Duration
    --SQL Error: ORA-01779: cannot modify a column which maps to a non key-preserved table
    UPDATE CourseOfferingsLastYear
        SET COURSE_DURATION = 4
            WHERE COURSE_ID = 3;

    --Can't modify Course Duration
    --SQL Error: ORA-01779: cannot modify a column which maps to a non key-preserved table
    UPDATE CourseOfferingsLastYear
        SET COURSE_DURATION = 4
            WHERE COURSE_COST = 300;

    --Can't modify Course Name
    --SQL Error: ORA-01779: cannot modify a column which maps to a non key-preserved table
    UPDATE CourseOfferingsLastYear
        SET COURSE_NAME = 'Testing'
            WHERE COURSE_ID = 3;

    --Can't modify Course ID
    --SQL Error: ORA-01779: cannot modify a column which maps to a non key-preserved table
    UPDATE CourseOfferingsLastYear
        SET COURSE_ID = 15
            WHERE COURSE_NAME = 'COURSE_NAME';

    --Can't modify Offering ID
    --ORA-00001: unique constraint (F021280L.PK_OFFERING) violated
    UPDATE CourseOfferingsLastYear
        SET OFFERING_ID = 15
            WHERE COURSE_COST = '700';

    --Can't modify STARTDATE
    --SQL Error: ORA-01779: cannot modify a column which maps to a non key-preserved table
    UPDATE CourseOfferingsLastYear
        SET COURSE_COST = 1500
            WHERE COURSE_ID = 1;

    --Works successfully
    UPDATE CourseOfferingsLastYear
        SET OFFERING_LOCATION = 'Testing'
            WHERE COURSE_ID = 1;
    --Works successfully
    UPDATE CourseOfferingsLastYear
        SET OFFERING_LOCATION = 'Testing NO CRITERIA';

    --Works successfully
    UPDATE CourseOfferingsLastYear
        SET INSTRUCTOR = 'Instructor test'
            WHERE COURSE_ID = 1;

    --Works successfully
    UPDATE CourseOfferingsLastYear
        SET INSTRUCTOR = 'Instructor test NO CRITERIA';
```

```sql

    --Works successfully
    UPDATE CourseOfferingsLastYear
        SET OFFERING_LOCATION = 'Testing'
            WHERE COURSE_ID = 1;
    --Works successfully
    UPDATE CourseOfferingsLastYear
        SET OFFERING_LOCATION = 'Testing NO CRITERIA';

    --Works successfully
    UPDATE CourseOfferingsLastYear
        SET INSTRUCTOR = 'Instructor test'
            WHERE COURSE_ID = 1;

    --Works successfully
    UPDATE CourseOfferingsLastYear
        SET INSTRUCTOR = 'Instructor test NO CRITERIA';

----- DELETES -------
-- Can't delete row as child record found
-- ORA-02292: integrity constraint (F021280L.FK_ATTENDEES_OFFERING) violated - child record found
DELETE FROM COURSEOFFERINGSLASTYEAR WHERE COURSE_ID = 1;

--Can delete this successfully
--There were no child records for its course ID (There were no students for this course)
--This also deleted it from OFFERING
DELETE FROM COURSEOFFERINGSLASTYEAR WHERE COURSE_ID = 5;

--Can delete this successfully
--There were no child records for its course ID (There were no students for this course)
--This also deleted it from OFFERING
DELETE FROM COURSEOFFERINGSLASTYEAR WHERE COURSE_ID = 4;

--Can delete this successfully
--There were no child records for its course ID (There were no students for this course)
--This also deleted it from OFFERING
DELETE FROM COURSEOFFERINGSLASTYEAR WHERE COURSE_ID = 3;

----- INSERT -------
--Cannot do this with all values
--SQL Error: ORA-01779: cannot modify a column which maps to a non key-preserved tabl
INSERT INTO COURSEOFFERINGSLASTYEAR VALUES('AWS',3,'4','700','10','01-02-21','Scranton','Michael Scott');

--Cannot do this with just values from COURSE table
--SQL Error: ORA-01779: cannot modify a column which maps to a non key-preserved table
INSERT INTO COURSEOFFERINGSLASTYEAR(COURSE_ID,COURSE_NAME,COURSE_DURATION,COURSE_COST)VALUES(1,'Leadership','1','300.00');

--Cannot do this like this; needs to be different
--SQL Error: ORA-01776: cannot modify more than one base table through a join view
INSERT INTO COURSEOFFERINGSLASTYEAR(OFFERING_ID,COURSE_ID,STARTDATE,OFFERING_LOCATION,INSTRUCTOR)VALUES(18,'4','03-JUN-21','New York','INSTRUCTOR');

--Success, COURSE_ID is null though so this would need to be done seperately
INSERT INTO COURSEOFFERINGSLASTYEAR(OFFERING_ID,STARTDATE,OFFERING_LOCATION,INSTRUCTOR) VALUES(18,'03-JUN-21','Offering Location1','TestInstructor');
--part 2 cannot be done, although trying to make a standard inssert to OFFERING without COURSE_ID
--SQL Error: ORA-01779: cannot modify a column which maps to a non key-preserved table e.g.
--INSERT INTO OFFERING (OFFERING_ID,STARTDATE,OFFERING_LOCATION,INSTRUCTOR) VALUES(25,'01-JUL-2021','Albuquerque','Walter White');
    UPDATE CourseOfferingsLastYear
    SET COURSE_ID = 4 WHERE OFFERING_ID = 18;
```

5b. Discussion

- Reason for error discussed in image/code
- In terms of ease of update, deletes were able to be done providing no child record was found, I.E., for deleting a row with a COURSE_ID of 1, it checked if the corresponding OFFERING row was referenced in a child table, I.E. ATTENDEES. Additionally, INSERTS were able to be done providing only values from the original table OFFERING were referenced, however COURSE_ID was set to null which was not useful and should not be done. This view was useful as a method to see the data but would not be as useful for modifying the data. However, in terms of security this view can be given to a user and besides the problem with COURSE_ID which could be set to not allow null, would allow a user to update

nonessential rows. Updates are difficult because it's a join between two tables which makes many of them unable to be done when referencing columns from both tables.

5c.

<u>Read access:</u>

```
GRANT SELECT ON COURSEOFFERINGSLASTYEAR TO
F021280LA;
```

	GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE	HIERARCHY
1	F021280LA	F021280L	COURSEOFFERINGSLASTYEAR	F021280L	SELECT	NO	NO

```
SELECT * FROM F021280L.COURSEOFFERINGSLASTYEAR;
```

SQL | All Rows Fetched: 15 in 0.043 seconds

	COURSE_NAME	COURSE_ID	COURSE_DURATION	COURSE_COST	OFFERING_ID	STARTDATE	OFFERING_LOCATION	INSTRUCTOR
1	Comptia	3	3	700	13	26-JAN-21	Scranton	Creed Bratton
2	Leadership	1	1	300	4	02-FEB-21	Pawnee	Ron Swanson
3	Leadership	1	1	300	5	09-FEB-21	Pawnee	April Ludgate
4	Leadership	1	1	300	6	16-FEB-21	Pawnee	Ben Wyatt
5	Project Management	2	2	500	7	01-MAR-21	London	Mark Corrigan
6	Project Management	2	2	500	8	08-MAR-21	London	Alan Johnson
7	Project Management	2	2	500	9	15-MAR-21	London	Jeremy Usborne
8	AWS	4	4	900	14	01-JUN-21	New York	Jake Peralta
9	AWS	4	4	900	15	08-JUN-21	New York	Rosa Diaz
10	AWS	4	4	900	16	15-JUN-21	New York	Amy Santiago
11	AWS	4	4	900	17	22-JUN-21	New York	Gina Linetti
12	Programming Fundamentals	5	5	1200	18	01-JUL-21	Albuquerque	Walter White
13	Leadership	1	1	300	1	12-NOV-21	Pawnee	Ron Swanson
14	Leadership	1	1	300	2	19-NOV-21	Pawnee	April Ludgate
15	Leadership	1	1	300	3	26-NOV-21	Pawnee	Leslie Knope

<u>Update access:</u>

```
GRANT UPDATE ON COURSEOFFERINGSLASTYEAR TO
F021280LA;
```

	GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE	HIERARCHY
1	F021280LA	F021280L	COURSEOFFERINGSLASTYEAR	F021280L	SELECT	NO	NO
2	F021280LA	F021280L	COURSEOFFERINGSLASTYEAR	F021280L	UPDATE	NO	NO

```
UPDATE F021280L.COURSEOFFERINGSLASTYEAR
  SET INSTRUCTOR = 'Instructor test NO CRITERIA';
```

```
--Works successfully
UPDATE F021280L.CourseOfferingsLastYear
    SET INSTRUCTOR = 'Instructor test NO CRITERIA';
```

Prohibit account:

REVOKE ALL ON COURSEOFFERINGSLASTYEAR FROM F021280LA;

```
REVOKE ALL ON COURSEOFFERINGSLASTYEAR FROM F021280LA;
```

Query Result

All Rows Fetched: 2 in 0.039 seconds

	GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE	HIERARCHY
1	F021280LA	SYS	DBMS_REDACT	SYS	EXECUTE	NO	NO
2	F021280LA	SYS	DBMS_LOCK	SYS	EXECUTE	NO	NO

Question 6.

```
BEGIN
  DBMS_REDACT.ADD_POLICY
   (object_schema => USER,
    object_name => 'STUDENT',
    policy_name => 'STUDENT_POLICY',
    column_name => 'CREDIT_CARD_NUMBER',
    function_type => DBMS_REDACT.PARTIAL,
    function_parameters => 'VVVVFVVVVFVVVVFVVVV,VVVV VVVV VVVV
VVVV,*,1,19',
    expression => '1=1');
END;
BEGIN
  DBMS_REDACT.ALTER_POLICY
   (object_schema => USER,
    object_name => 'STUDENT',
```

```
    policy_name => 'STUDENT_POLICY',
    action => DBMS_REDACT.ADD_COLUMN,
    column_name => 'DATEOFBIRTH',
    function_type => DBMS_REDACT.PARTIAL,
    function_parameters => 'm5d5Y',
    expression => '1=1');
END;
```

For ease of reading:

```
BEGIN
  DBMS_REDACT.ADD_POLICY
   (object_schema => USER,
    object_name => 'STUDENT',
    policy_name => 'STUDENT_POLICY',
    column_name => 'CREDIT_CARD_NUMBER',
    function_type => DBMS_REDACT.PARTIAL,
    function_parameters => 'VVVVFVVVVFVVVVFVVVV,VVVV VVVV VVVV VVVV,*,1,19',
    expression => '1=1');
END;

BEGIN
  DBMS_REDACT.ALTER_POLICY
   (object_schema => USER,
    object_name => 'STUDENT',
    policy_name => 'STUDENT_POLICY',
    action => DBMS_REDACT.ADD_COLUMN,
    column_name => 'DATEOFBIRTH',
    function_type => DBMS_REDACT.PARTIAL,
    function_parameters => 'm5d5Y',
    expression => '1=1');
END;
/
```

Question 7.

Introduction

With databases potentially storing valuable financial data about customers, it's important to secure your database against would-be attackers. Cyber-attacks may choose to capitalize on weaknesses in a system as opposed to social engineering, for example brute forcing a system through automated trial and error. For this reason, a good authentication system is important for developing a secure database system.

Authentication Methods

Passwords

Database authentication uses information stored in the given database to secure a system I.E. a username and password. The first password can be traced back to the Compatible Time-Sharing System (CTSS) at MIT, where Fernando Corbató noticed that the CTSS system did not offer a way to secure private files per user. As a result of this, he developed a novel idea to use a password, stored in plain text. Since then, various addendums to password security have been made, including encryption, a method of scrambling a password so it's unusable to hackers. Oracle does this by encrypting passwords using the Advanced Encryption Standard (AES) when connecting to a database session, however by default, network trace files still store passwords in plaintext. Authentication has various methods, including password, biometrics, and group policies. In Oracle, password management features can include requiring a specific password complexity, limiting the number of failures allowed between successful logins, limiting the duration and number of password changes in which a password may not be reused, setting a maximum password lifetime and the number of days the previous password can be used after a password has expired. In concern of a data breach, a database administrator can also forcefully expire a user account to require them to change their password when they log in next. The commands for these management features are described in Table 1.

Password Management Feature	Oracle Command
Limit the number of failures between successful logins	CREATE PROFILE user LIMIT FAILED_LOGIN_ATTEMPTS {NUMBER}
Limiting the duration of a password	CREATE PROFILE user LIMIT PASSWORD_LIFE_TIME {DURATION}
Limiting the number of password changes required before a password can be reused	CREATE PROFILE user LIMIT PASSWORD_REUSE_TIME {NUMBER}
Time interval in which a password cannot be reused	CREATE PROFILE user LIMIT PASSWORD_REUSE_TIME {DURATION}
The grace period in which a previous password can still be used	CREATE PROFILE user LIMIT PASSWORD_GRACE_TIME {DURATTION}
Forcefully expire a user password	ALTER USER jackfletcher PASSWORD expire;

Additionally, Oracle has a built-in script to assign a minimum password complexity known as 'utlpwdmg.sql'. This sets the default parameters set out below.

- Minimum password length of 4
- Password is not the same as User ID
- Password has at least one alpha, numeric and special character.
- The password does not match common words that can be used in a dictionary attack.
- The password differs from the previous password by three characters.

Database level authentication

One method of doing this, is using database level authentication. To use database authentication, a user must be associated with a corresponding username and password. In Oracle, this can be done using the command mentioned in figure 1.

```
CREATE USER jackfletcher IDENTIFIED BY test;
```

Figure 1: Creating a user with password in Oracle 11C

Creating a user alone does not grant users access to a database however, for this they need a system privilege known as 'CREATE SESSION'.

This allows database administrators to control access to the database to a specific subset of users using well known methods and can be enhanced using the password management features mentioned above to increase security. Additionally, this is an easy authentication method to use when there is a relatively small number of users.

External authentication

External authentication is the use of third-party software to authenticate users of a system. For example, in Oracle, an Oracle database Is used for user account maintenance, however an external service such as an operating system (OS) or Kerberos is used for password administration and user authentication.

To create a user, a variation of the typical statement must be used, described in figure 2. Note, a password is not required as external authentication manages the password externally.

```
-- Creating a local account external user
CREATE USER jackfletcher IDENTIFIED EXTERNALLY;

-- Creating a domain external user
CREATE USER "DOMAINNAME\jackfletcher" IDENTIFIED EXTERNALLY;
```

Figure 2 Creating an External User

If using external authentication, a password is not used. For (OS) solutions, Oracle requires the 'OS_AUTHENT_PREFIX' parameter be set in the init.ora file. By default, this is set to OP$, which is then prefixed to the users account name and compared to the database to check whether it is a valid user. This does however require a secure connection to be made, which can preclude using a shared server configuration. This can be removed; however, this is poor practice and should not be done.

Using external authentication allows more choices of authentication methods, such as biometric scanners, single sign on, or an operating system login.

Global authentication and authorization

Global authentication and authorization use a singular microservice to handle all authentication and authorization, for example Microsoft Active Directory. Users and administrators are identified in the database as global users which signifies that they are authenticated by an external directory service. This allows creation of enterprise roles and users which can be thought of as a container object for one or more global roles. When a user logs into a database, the database checks with the microservice as to whether the user's enterprise roles contain a global role for the database and if so, enables the global role for the user.

Global authentication and authorisation allow a central management of users and permissions across the business, which makes it easier to make global changes to a role's permissions. Additionally, this correlates to ease of use for the user. Due to the single point of access, this can allow for single sign on among various linked services, so the user only needs to learn one login.