Homework 1

Your Name: Naga Satya Silpa Annadevara

Student ID: A20517818

---

1. Understanding Concepts related to DB

a). Compare and introduce the differences between relational data models and ER data models (10 points)

Relational Data model:
Relational data model uses Tables(entities) to represent data and the relation between this data. These tables are referred to as 'Relations' In relational model.
Tables can have an unlimited number of rows, but they must have a specific number of columns that can link to each other by using Primary keys and foreign keys.

Each row in a table is known as a "Tuple," and it holds all the details about a certain table item. Records are collections of tuples; hence the relational model is sometimes known as a record-based model.

A table's columns are referred to as its attributes since they describe the table's characteristics (relation). Each attribute must have a domain that specifies the kind of data it may hold.

All three types of relations—one to many, many to one, and many to many—are often accommodated by relational data architectures.

ER Data model:
Entity relation models are used for Graphical representation of relation models.

A collection of entities, often known as real world objects, and the relationships between those entities may be described as the E-R Model. There should be no identikit entities. The database's conceptual view is provided through the E-R Model.

Entity set, relationship set, and characteristics combine to form the E-R Model. In this scenario, all entities together form an entity set, all relationships between entities together form an entity relationship set, and the attribute represents the attributes of the entities.

Relational Data Model	ER Data Model
1. It is used to display a group of tables as well as the connections between them.	1. It is used to describe a group of things called Entities (which are also called Tables) as well as the connections between them.
2. It is Table specific	2. It Is entity specific
3. This model does not cover cardinality mapping.	3. The mapping of Cardinalities is described by this model. Cardinality is the measure of how distinct a row of data items is.
4. It uses domain, attributes, and tuples to represent components.	4. Entity, Entity Type, and Entity Set are represented as components.
5. It is a linear data model without any modeling diagrams.	5. It uses a mapping diagram for its representation.
6. In Relational model, it is less difficult or more complex to understand the relation between tables.	6. Understanding the relationships between entities is simpler using the E-R Model.
7. It is the representational or implementation model.	7. This model is theoretical or high-level.

b). Explain the differences: strong entity, weak entity, associative entity (10 points)

Entity:
An Entity is nothing but a table which consists of a set of entity instances (rows) and Attributes (columns).
Example: Student Entity, Course Entity, Department Entity.

Strong Entity	Weak Entity	Associative Entity
1. Strong Entity is not dependent on any other entity. They are Independent.	1. Weak entity cannot exist on its own, so it depends on strong entity	1. The associated entity is also a weak entity since it lacks an independent existence.
2. A primary key exists for every strong entity	2. A weak entity is one that can't be uniquely recognized by only its qualities; as a result, it must combine a foreign key with those properties to get a primary key.	2. An associating entity is one that is inserted between two other entities to break a M:N link.
3. A single rectangle serves as a strong entity's indicator.	3. With a double rectangle, a weak entity is indicated.	3. A single rectangle serves as an associative entity's indicator.

4. Strong entities in ERD are represented by single line boxes.	4. Weak entities are represented by double line boxes in ERD.	
5. A single diamond denotes the relationship between two strong entities.	5. A double diamond is used to symbolize the relationship between two opposing entities—one powerful and one weak.	
6. Strong entity might or might not be fully involved in the connection.	6. The identifying relationship depicted by the double line is always fully participated in by the weak entity.	6. It is used to depict M: N connections more accurately. It can aid in the conversion of M: N connections to multiple 1:M relationships.

c). Why we use index? What are the differences between primary key and unique index? (10 points)

Index:
-An index is a schema object that holds an entry for each value that appears in the table or cluster's indexed column(s) and allows for direct, quick access to rows.
-Indexes can help enhance SQL query performance.
- It may be set on a single attribute or a composite of attributes. Allowing duplicate and NULL values

Primary key:
It is a constraint placed on a table's attribute to prevent duplicate rows and to prevent Null values from being present. Every row value for every characteristic is required to be unique. It serves as a unique identifier for every record.

Unique Index:
Unique indexes make sure that no two rows of data in a database have the same key values, which helps to ensure data integrity.

Primary Key	Unique Index
1. a Primary key is a chosen candidate key that is used to distinctly identify any specific row;	1. An index that can identify every row that is distinct, such as the unique values in a unique index, designed for rapid searches or queries
2. One can make the primary key a unique index. If a primary key has already been generated, the database will set it as a unique index by default.	2. Nonetheless, a unique index need not be a main key.

3. There should only be one primary key per table.	3. However, a table may have more than one unique index.
4. Primary key don't allow Null values.	4. Unique Index allow Null values.
5. Primary key don't allow duplicate values	5. Unique Index allows Duplicate values but only 1.

3. (70 points) Given the following tables:
Customer Table [assume email is unique for each customer]
CustomerID, Name, Age, Gender, Email
Order Table
OrderID, CustomerID, OrderDate, OrderTotal, OrderPayment
OrderItems Table [describe items in an order. Note that there may be several items in a single order]
OrderID, ItemID, ItemQuantity, ItemUnitPrice

1). (15 points) For each table, list the following keys: super key, candidate key, primary key, secondary key, foreign key

Customer Table:

Super key:

Customer ID
Customer ID + Name
Customer ID + Age
Customer ID + Gender
Customer ID + Email
Customer ID + Name + Age + Gender +Email
Customer ID + Name + Age
Customer ID + Name + Gender
Customer ID + Name + Email
Customer ID + Age + Gender
Customer ID + Age + Email
Customer ID + Gender + Email
Email
Email + Customer ID
Email + Name
Email + Age
Email + Gender
Email + Name + Age
Email + Name + Gender
Email + Age + Gender

Customer ID + Email

Candidate key:
Email,
Customer ID

Primary key:
Customer ID

Secondary key:
Email

Foreign key:
None

Order Table:

Super key:
Order ID
Order ID + Customer ID
Order ID + Customer ID + Order Date
Order ID + Customer ID + Order Date + Order Total
Order ID + Customer ID + Order Date + Order Total + Order Payment
Order ID + Order Date
Order ID + Order Date + Order Total
Order ID + Order Date + Order Total + Order Payment
Order ID + Order Total
Order ID + Order Total + Order Payment
Order ID + Order Payment
Order ID + Order Payment + Customer ID

Candidate key:
Order ID

Primary key:
Order ID

Secondary key:
None

Foreign key:
Customer ID

OrderItems Table:

Super key:
Item ID
Item ID + Order ID
Item ID + Order ID + Item Quantity
Item ID + Order ID + Item Quantity + Item Unit Price
Item ID + Item Quantity
Item ID + Item Quantity + Item Unit Price
Item ID + Item Unit Price
Item ID + Item Unit Price + Odder ID
Order ID
Order ID + Any Other Attribute

Candidate key:
Item ID

Primary key:
Item ID

Secondary key:
None

Foreign key:
Order ID
Item ID

2). (20 points) List relationships between Customer and Order, Order and OrderItems, and explain why. Note: you do not need to draw the diagram

Relationship between Customer and Order:

Customer → Order

from left to right: 1 customer can have many orders. So the relationship between Customer and Order from left to right is 1(mandatory) : M(optional)

From right to left: 1 Order can be handed over to only 1 Customer. So the relationship between Order and Customer from right to left is 1(mandatory) : 1

So, the relation between Customer and Order is 1(mandatory) : M(optional)

Relationship between Order and Order Items:

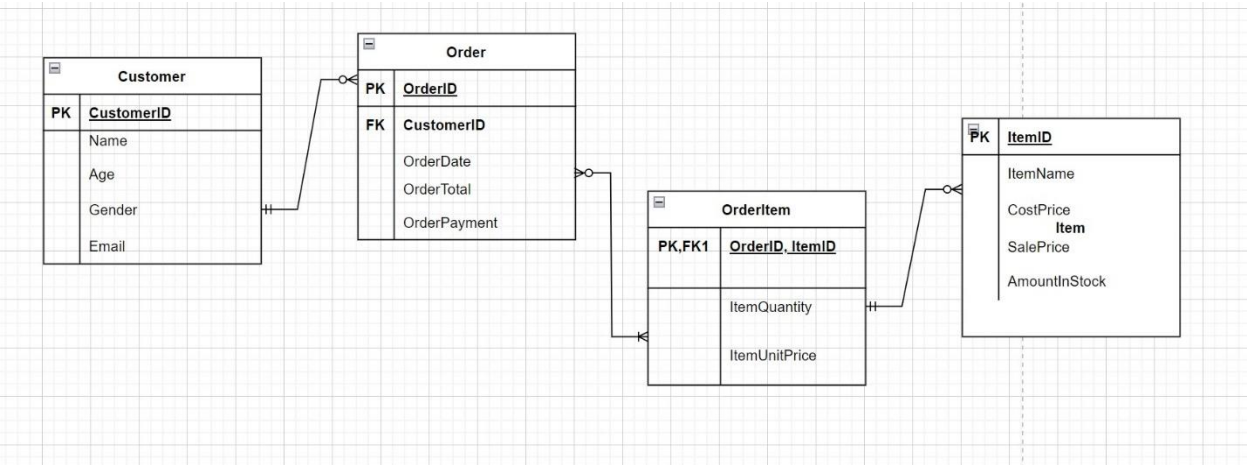
Order → Order Items

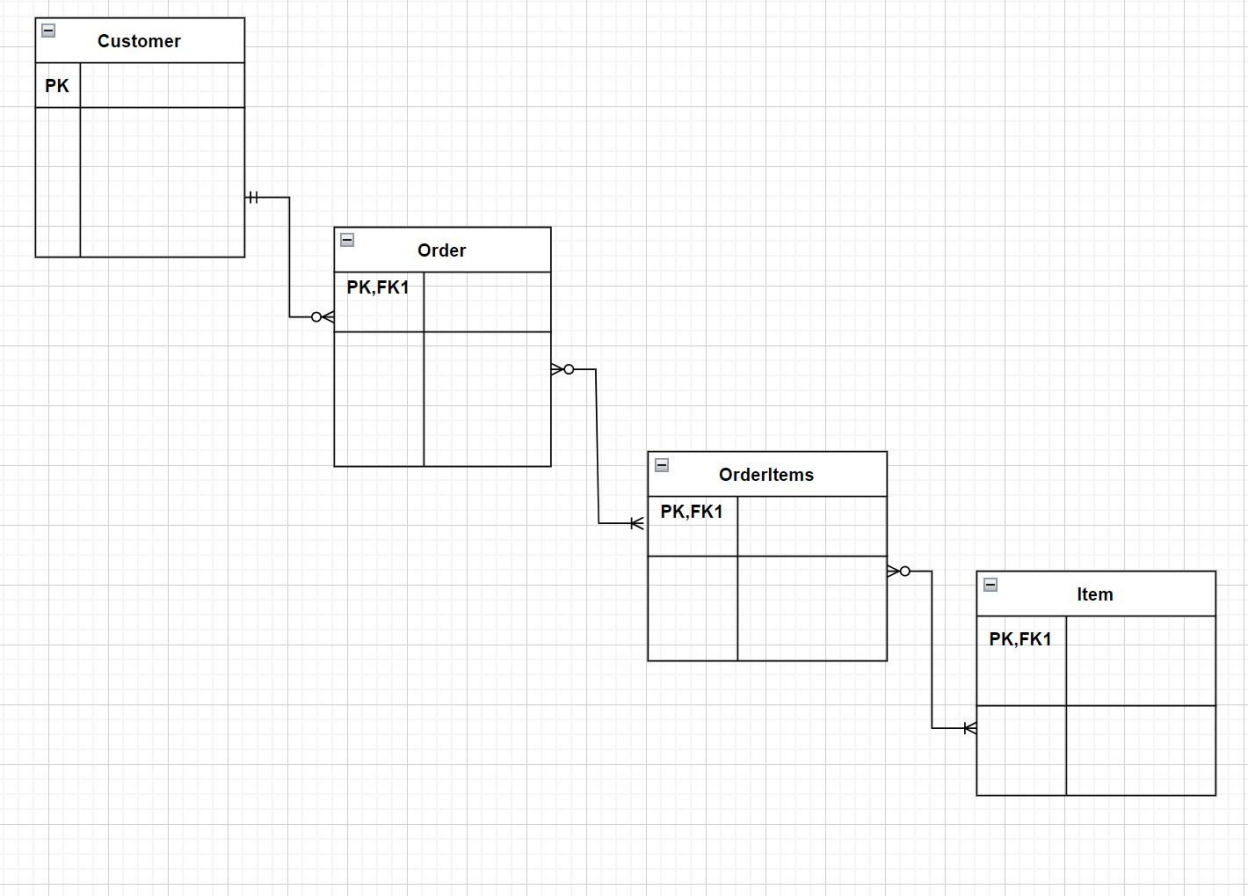
From left to right: 1 order can have many order items. So, the relationship between order and order items from left to right is 1(mandatory) : M(optional)
From right to left: 1 item can be in many orders. So the relation between order and order items from right to left is 1: M

So, the relation between order and order items is M(optional) : N(mandatory)

3). (35 points) Assume we have another table – Item with attributes (ItemID, ItemName, CostPrice, SalePrice, AmountInStock), complete DB conceptual design (based on all of these entities) by using ERD with Crow's foot notations. Note: you can use draw.io to draw ERD and you must provide ERD images (PNG or JPG) in the document

DB Conceptual design:





My Grading for this assignment is 91.

Incorrect answers:

HW1

Q3. 1. -5 Incorrect keys provided for the below tables

OrderItems Table : Foreign Key, Super Key, Secondary Key, Primary Key, Candidate Key

Q3. 2. -2 Minimum cardinality (constraints) for order and OrderItem is incorrect

Q3. 3. -2 Crow's foot notations between order and orderitems is Incorrect.

Homework 2

Your Name: Naga Satya Silpa Annadevara

Student ID: A20517818

---

1. Normalization [20 points]

The table shown below displays the details of the roles played by actors/actresses in films.

filmNo	fTitle	dirNo	director	actorNo	aName	role	timeOnScreen
F1100	Happy Days	D101	Jim Alan	A1020	Sheila Toner	Jean Simson	15.45
		D101	Jim Alan	A1222	Peter Watt	Tom Kinder	25.38
		D101	Jim Alan	A1020	Sheila Toner	Silvia Simpson	22.56
F1109	Snake Bite	D076	Sue Ramsay	A1567	Steven McDonald	Tim Rosey	19.56
		D076	Sue Ramsay	A1222	Peter Watt	Archie Bold	10.44

(a) [5 points] Describe why the table shown below is not in first normal form and second normal form.

ANSWER:

1NF REQUIREMENT: There should be NO MULTI-VALUED ATTRIBUTES in Entities if it has to be in first normal form, all the rows should be unique with the Primary key identified. But the table above has multiple values in the fields such as dirNo, director, actorNo, aName, role, timeOnScreen. So is thereason the tables is not in 1NF.

2NF REQUIREMENT: A table should meet the requirements of 1NF + the table should have NO PARTIAL DEPENDENCIES in it. But the table above didn't meet the 1NF requirements + the table has partial dependencies in it such as – actorName is dependent on actorNo. And actorNo is part pf candidate key if filmNo and actorNo together taken as a candidate key.

(b) [15 points] Re-design the tables to make them meet BCNF; you do not need to draw ERD, but you need to show the final design by drawing data tables in Word document and explain why it meets BCNF. Just List the tables, attributes and keys. Use * to indicate Primary Key, use # to indicate foreign key.
Example: Student (StudentID *, Name, DeptID #)

Director

dirNo * (PK)	director
D 101	Jim Alan
D 076	Sue Ramsay

FILM

filmNo. * (PK)	fTitle	dirNo # (FK)
F 1100	Happy Days	D 101
F 1109	Snake Bite	D 076

ACTOR

actorNo * (PK)	aName
A1020	Sheila Toner
A1222	Peter Watt
A1222	Peter Watt
A1567	Steven McDonald

ROLE

Role ID * (PK)	filmNo. # (FK)	actorNo. # (FK)	role	timeOnScreen
1	F 1100	A1020	Jean Simson	15.45
2	F 1100	A1222	Tom Kinder	25.38
3	F 1100	A1020	Silvia Simpson	22.56
4	F 1109	A1567	Tim Rosey	19.56
5	F 1109	A1222	Archie Bold	10.44

Explanation: First, second, third, and BCNF form compliance are all met by the database design described above. There are no partial or transitive dependencies in any table, nor are there multiple valued attributes. Every table has One candidate key .

2. Identify issues in DB design [25 points]

Assume that the primary key of this relation in the data below consists of two components: Author's ID (AID) and book number (BNbr). The relation includes data regarding authors, books and publishers. In addition, it tells what an individual author's per book royalty amount is in the case of multi-authored books

TABLE 4-9 Author Book Royalties

AID	ALname	AFname	AInst	BNbr	BName	BPublish	PubCity	BPrice	AuthBRoyalty
10	Gold	Josh	Sleepy Hollow U	106	JavaScript and HTML5	Wall & Vintage	Chicago, IL	$62.75	$6.28
				102	Quick Mobile Apps	Gray Brothers	Boston, MA	$49.95	$2.50
24	Shippen	Mary	Green Lawns U	104	Innovative Data Management	Smith and Sons	Dallas, TX	$158.65	$15.87
				106	JavaScript and HTML5	Wall & Vintage	Indianapolis, IN	$62.75	$6.00
32	Oswan	Jan	Middlestate College	126	Networks and Data Centers	Grey Brothers	Boston, NH	$250.00	$12.50
				180	Server Infrastructure	Gray Brothers	Boston, MA	$122.85	$12.30
				102	Quick Mobile Apps	Gray Brothers	Boston, MA	$45.00	$2.25

1). Identify the normal forms in the DB design above, and explain why [10 points]

ANSWER:

The Table has multi-valued attributes in it(Alnst, BNbr, BName). So, it is not in First Normal Form(1NF)

The Table has Partial dependencies in it where BName is dependent on BNbr(bookNumber). So, It is not in Second Normal Form(2NF)

The Table had Transitive dependencies in it where Bname is dependent on AuthorName based on BNbr.

So the table above don't have any normal forms in it.

3). Take actions (if necessary) to convert the relations into the 3rd normal form. Show your steps how did you fix the issues [15 points]

ANSWER:

The table above is not in 3NF. To fix this:

First the table need to get converted into 1NF: Remove all the multi value attributes by creating a separate records for each repeated values.

Then the table needs to get converted into 2NF: The table is in 1NF now + **Remove** all the **partial dependencies** that are existing in the table.

Here, AID and BNbr together is the primary as it is already mentioned in the question. So AuthBRoyality is the only non-key attribute that is dependent on the entire primary key (AID+BNbr) So AuthBRoyality will stay with AID and BNbr in the original table.

AUTHBRoyality

AID (foreign key)
BNbr (foreign key)
AuthBRoyality

ALname , AFName, AInst are(non-key attributes) that are partially dependent on only AID (part of the primary key) so remove ALname, AFName and AInst from the original table and create a separate table along with AID & name that NEW table as AUTHOR table.

Set AID in the new table as Primary key and as a foreign key in the old table.

AUTHOR

AID (Primary key)
ALname
AFNAme
AInst

And, BName, BPublish, PubCity, BPrice are partially dependent on BNbr, create a NEW TABLE with all of these and remove (BName, BPublish, PubCity, BPrice) from the old table.Name that NEW TABLE as BOOK table

Set BNbr as a primary key in the new table and as a foreign key in the original table.

BOOK

BNbr (primary key)
BName
BPublish
PubCity,
BPrice)

There is transitive dependency in BOOK table. And the design is in 2NF but not in 3NF.

So The table need to get converted in to 3NF: Remove transitive dependencies in the BOOK table in which PubCity is indirectly dependent on BNbr through BPublish.

So create a new table with BPublish & PubCity and name that table as PUBLISHER

Set BPublish as a Primary key in new table.

Remove Pubcity in Older table.

Set BPublish as foreign key in older table.

Hence the design we get :

BOOK

BNbr (primary key)
BName
BPublish (foreign key)
BPrice

PUBLISHER

BPublish (primary key)
PubCity

Here is the final design in 3NF:

AUTHORBOOKROYALTY(AID*, BNbr*, AuthBRoyalty)

AID references AUTHOR(AID)

BNbr references BOOK(BNbr)

AUTHOR(AID*, ALName, AFName, AInst)

BOOK(BNbr*, BName, BPublish, BPrice)

BPublish references PUBLISHER(BPublish)

PUBLISHER(BPublish*, PubCity)

3. Database Design [55 points]

Complete conceptual and logical DB design, and show your answers step by step
Note: do not need to draw ERD, just show tables, attributes, and keys
Example: Student (StudentID *, Name, DeptID #)

Descriptions:
IIT wants to build a DB to collect faculties' publications.

- **Faculties may be from different departments.**
- **A publication should have title, publication date, and publication venue (such as a conference or a journal). There are only two publication types: conference publications or Journal publications. A publication may have more than one authors.**
- **A full-time faculty is expected to have at least 2 publications per year.**

1). Complete conceptual DB design, by listing entities and indicating their relationships [15]

ANSWER:

The entities we have are:

1. Faculty
2. Department
3. Publication (Title, p_date, p_venue, p_type)

Relationship between faculty and Department:

faculty → Department

from left to right: 1 faculty can be in many departments. So the relationship between faculty and department from left to right is 1(mandatory) : M(optional)

From right to left: 1 department should have at least 1 faculty. So the relationship between department and faculty from right to left is 1(mandatory) : 1

So, the relation between faculty and department is 1(mandatory) : M(optional)

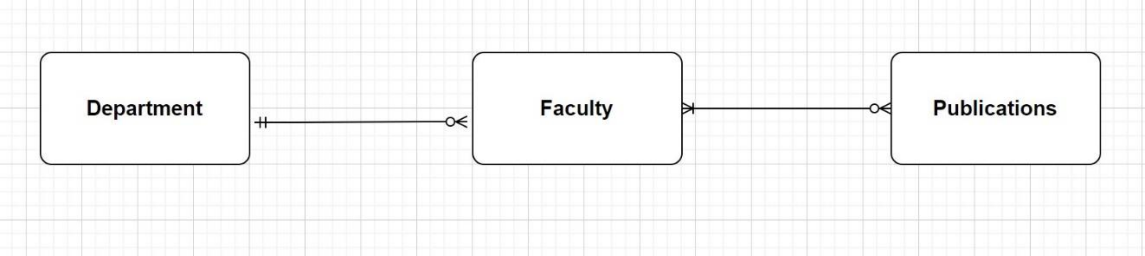
Relationship between faculty and Publication:

faculty → Publication

from left to right: 1 faculty can have many publications or No publications. So the relationship between faculty and department from left to right is 1(mandatory) : M(optional)

From right to left: 1 publication should have at least 1 faculty or many. So the relationship between department and faculty from right to left is 1(mandatory) : Many (optional)

So, the relation between faculty and Publication is M(mandatory) : N(optional)
(Many mandatory: many optional)



2). Complete logical DB design by assigning attributes and PK/FK, use normalization to make sure your design meets at least 3NF [40]

ANSWER:

Logical DB Design:

Assigning attributes to the entities and assigning PK & FK in each entity:

Faculty (FacultyID -primary key, DepartmentID -foreign key, FName, LName, EmailId, Employement_Status)

Department (DepartmentID- primary key, Dept_Name)

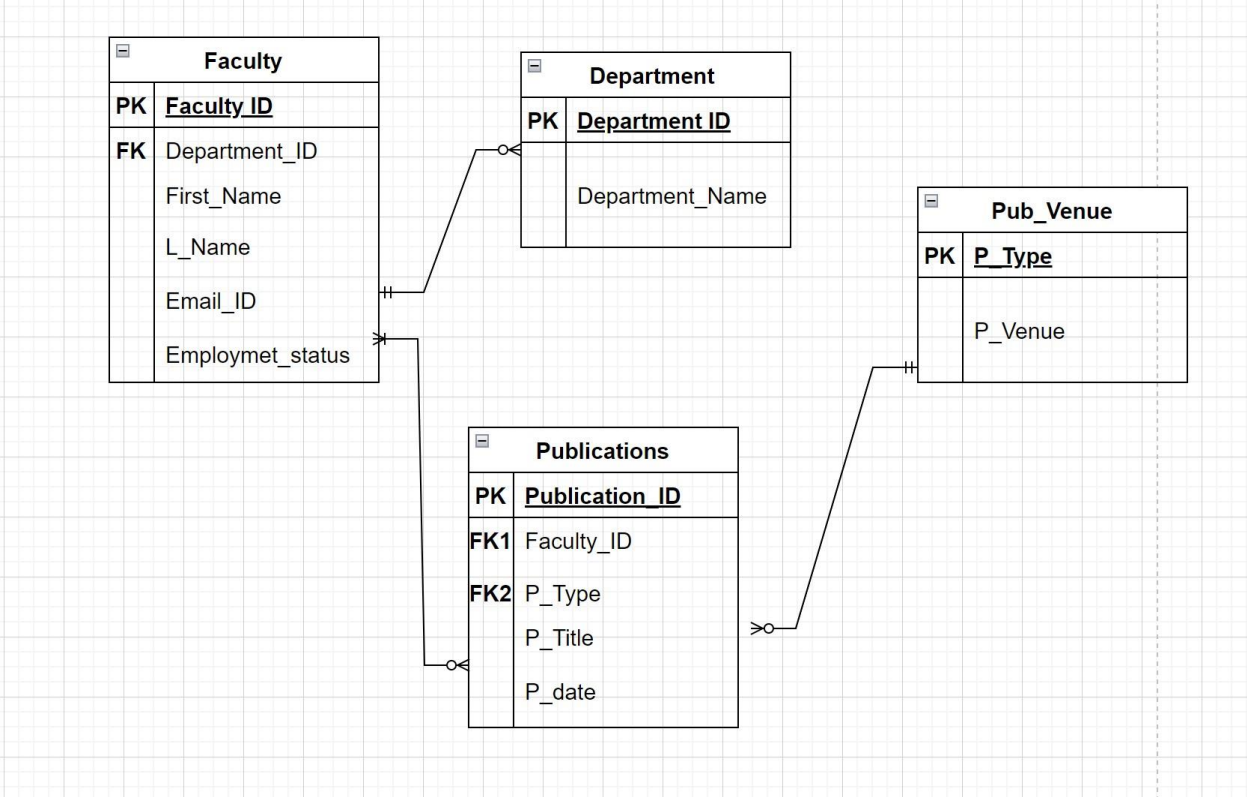
Publication (p_ ID -primary key, Title, p_date, p_venue, p_type, FacultyID -foreign key)

Here we can see a transitive dependency in Publication table where, P_Type(publication_type) is dependent on Pub_ID and P_Venue(Publication_Venue) is dependent on P_Type

so, we break the table as follows:

Publications(Pub_ID - primary key, FacultyID – foreign key, P_Type – foreign key, Title, P_Date)

Pub_Venue(P_Type – primary key, P_Venue)



Result & feedback: I got 93.

3. 1. -2 Some relationship between entities are missing such as author and venue.

3. 2. -5 Critical entities such as author is missing from your design It's possible for a single paper to have multiple authors, and these authors may not necessarily be affiliated with IIT.

Entity Venue is not designed correctly (Authors can publish work in conference or journals. For a same journal or conference, a same author may have several publications in these venues).

You can't determine whether an author of a particular publication is a faculty

Full time status of a faculty is not implemented in the DB design

Homework 3

Your Name: Naga Satya Silpa Annadevara

Student ID: A20517818

Q1. Look at the data table below. Convert it to BCNF. Show the steps for 1NF, 2NF, 3NF, BCNF. In each step, explain why it is in or not in a specific normal form, and discuss your solutions to transform them into the corresponding normal form [20 points]

TABLE 4-6 Parking Tickets at Millennium College

Parking Ticket Table

St ID	L Name	F Name	Phone No	St Lic	Lic No	Ticket #	Date	Code	Fine
38249	Brown	Thomas	111-7804	FL	BRY 123	15634	10/17/2018	2	$25
						16017	11/13/2018	1	$15
82453	Green	Sally	391-1689	AL	TRE 141	14987	10/05/2018	3	$100
						16293	11/18/2018	1	$15
						17892	12/13/2018	2	$25

ANSWER:

1NF Requirement:

1. There should be no multi-valued attributes. Each column should contain single values only.
2. Primary key should be identified.

Solution:
The table above is not in 1NF because the attributes (Ticket#, Date, Code, Fine) have multi-values in them. So, create a new table by adding some more rows by eleminating multi-values in one row and set Ticket# as a primary key.

Hence the new design that is in 1NF is as follows:

St ID	L Name	F Name	Phone No	St Lic	Lic No	**Ticket # (PK)**	Date	Code	Fine
38249	Brown	Thomas	111-7804	FL	BRY 123	15634	10/17/2018	2	$25
38249	Brown	Thomas	111-7804	FL	BRY 123	16017	11/13/2018	1	$15
82453	Green	Sally	391-1689	AL	TRE 141	14987	10/05/2018	3	$100
82453	Green	Sally	391-1689	AL	TRE 141	16293	11/18/2018	1	$15
82453	Green	Sally	391-1689	AL	TRE 141	17892	12/13/2018	2	$25

2NF Requirement:

- The entity must be in 1NF first.
- 2. Second, there should be no Partial dependency. (X -> Y (X determines Y (or) Y is partially dependent on X) and X is only part of the candidate key, not the entire candidate key.

Solution:

- Create a new table with X and Y(Y can a single attribute or a combination of attributes).
- Set X as PK in new table, and as FK in old table.
- Remove Y from the old table.

The table above is in 1NF but not in 2NF because if we set (St ID and Ticket #) together as a candidate key, (L name, F name, Phone No, St Lic, Lic No (Y)) are partially dependent on St Id(X). To resolve this, follow the steps below

- Create a new table with all these attributes, X and Y
- Set St Id(X) as an FK in original table and PK in the new table.
- Remove (Y) (L name, F name, Phone No, St Lic, Lic No) in the original table.
- All the attributes are dependent on Ticket #.
- Ticket #, Code, Date, Fine, St Id(FK) will stay in original table.

Hence the new design that is in 2NF is as follows:

Ticket# (PK)	Date	Code	Fine	StID (FK)
15634	10/17/2018	2	25$	38249
16017	11/13/2018	1	$15	38249
14987	10/05/2018	3	$100	82453
16293	11/18/2018	1	$15	82453
17892	12/13/2018	2	$25	82453

StID (PK)	Lname	Fname	Phone	St Lic	Lic no
38249	Brown	Thomas	111-7804	FL	BRY123
82453	Green	Sally	391-01689	AL	TRE 141

3 NF Requirement:

- The table must be in 2NF.
- 2. There should be no Transitive dependency in the table. (If X -> B ->Y (X determines B and B determines Y), and B is a non-key attribute, then Y is transitively dependent on X via B.)

Solution:

- Create a new table with B and Y (Y can a single attribute or a combination of attributes)

- Remove Y from the old table.
- Set B as PK in new Table and as a FK in old table.

The design above is in 2NF but not in 3NF because there is Transitive dependency in table 1 where Fine (Y) is dependent on Code(B) and Code is dependent on Ticket # (X).
To resolve this, follow the steps below:

- Create a new table with Code(B) and Fine(Y)
- Remove Fine(Y) from the old table.
- Set Code(B) as PK in new table and as a FK in old table.

There is Transitive dependency in table 2 as well, where St Lic(Y) is dependent on Lic No(B) and Lic No is dependent on St ID(X).
To Resolve this, follow the steps below:

- Create a new table with Lic No(B) and St Lic(Y)
- Remove St Lic(Y) from the old table
- Set Lic No(B) as PK in new table and as a FK in old table

Hence the new design that is in 3NF is as follows:

Ticket# (PK)	Date	Code (FK)	St ID (FK)
15634	10/17/2018	2	38249
16017	11/13/2018	1	38249
14987	10/05/2018	3	82453
16293	11/18/2018	1	82453
17892	12/13/2018	2	82453

Code (PK)	Fine
1	$15
2	$25
3	$100

StID (PK)	Lname	Fname	Phone	Lic No (FK)
38249	Brown	Thomas	111-7804	BRY123
82453	Green	Sally	391-01689	TRE 141

St Lic	Lic No (PK)
FL	BRY123
AL	TRE 141

BCNF Requirement:

- The table must be in 3NF
- Every determinant must be a candidate key (in other words only full dependency is allowed)

the table must meet one of the following:

- Each table must have 1 candidate key (single attribute or a combination of attributes)
- Each table must have multiple candidate keys but each candidate key must have only 1 attribute.

Solution:
the table must meet one of the following:

- It must be in 3NF
- Each table must have 1 candidate key (single attribute or a combination of attributes)
- Each table must have multiple candidate keys but each candidate key must have only 1 attribute.

The design above is in 3NF but in BCNF, Because in table 1 there are multiple candidate keys (Ticket# is already a CK and Code + St ID together forms as a CK which is BCNF rule violation)
So, create separate table.
Break down the tables even more into smaller ones.

Hence the BCNF design is as follows:

Ticket# (PK)	Date	Code (FK)
15634	10/17/2018	2
16017	11/13/2018	1
14987	10/05/2018	3
16293	11/18/2018	1
17892	12/13/2018	2

Ticket# (PK)	Date	StID (FK)
15634	10/17/2018	38249
16017	11/13/2018	38249
14987	10/05/2018	82453
16293	11/18/2018	82453
17892	12/13/2018	82453

Code (PK)	Fine
1	$15
2	$25
3	$100

StID (PK)	Lname	Fname	Lic No (FK)
38249	Brown	Thomas	BRY123
82453	Green	Sally	TRE 141

St Lic	Lic No (PK)
FL	BRY123
AL	TRE 141

StID (PK)	Phone
38249	111-7804
82453	391-01689

Q2. Given the tables below, figure out super keys, candidate keys, primary keys, secondary keys and foreign keys [10 points]

EmpID	Name	BirthYear	Gender	Address	SSN	DeptID	Email

DeptID	Name	Address	DeptHead

ANSWER:

Table 1 (Employee table):

Super key: EmpID , SSN, Email

- EmpID with any other attributes.
- SSN with any other attributes.
- Email with any other attributes.
- All the 3 super keys together
- All the attributes together

Candidate key: EmpID , SSN, Email

Primary key: EmpID

Secondary key: SSN, Email

Foreign key: DeptID

Table 2 (Department table):

Super key: DeptID with any other attribute

Candidate key: DeptID

Primary key: DeptID

Secondary key: None

Foreign key: None

Q3. Below is the diagram for a university dinning service [20 points]

In this scenario, we assume that the work schedule is defined for each specific event, and one staff may work for different positions on different events.

a). identify the relationships for STAFF-STAFF and explain why.

ANSWER:

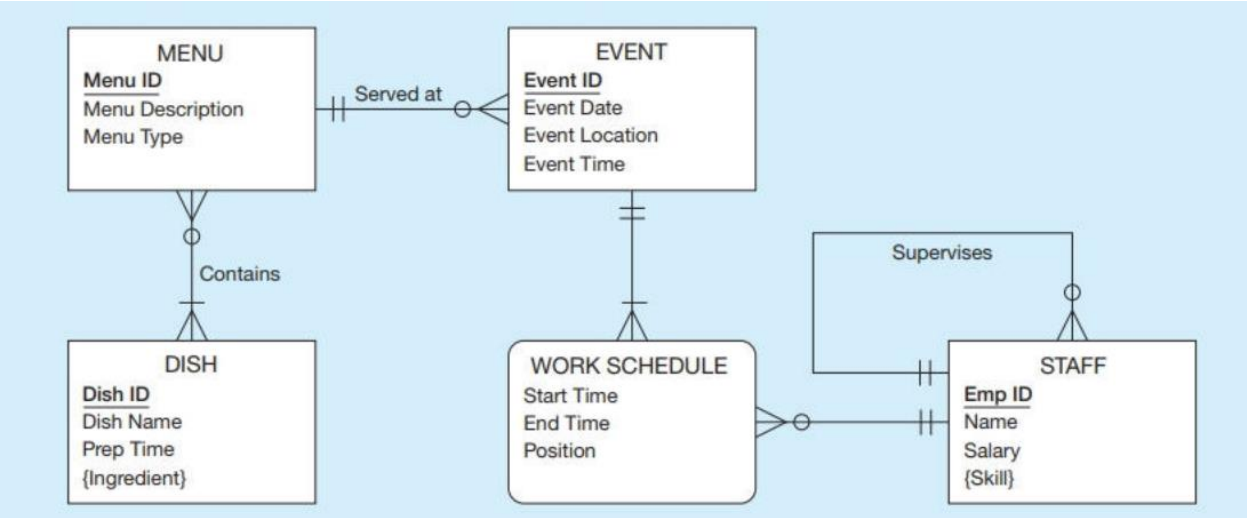
The relation for STAFF-STAFF is "Recursive relationship" is one in which a relationship can exist between occurrences of the same entity set.

It is because, In STAFF entity, there might be a MANAGER or SUPERVISOR who is also an Employee/Staff with EmpID that SUPERVISES other employees. So, here the MANAGER or SUPERVISOR (employee/Staff) details will also be held in staff entity.

An Employee/Staff(MANAGER or SUPERVISOR) may supervise many employees/Staff, and each Employee/Staff is managed by at least 1 employee/staff(MANAGER/SUPERVISOR).

So, The relation is 1(mandatory) : M(optional) (unary relationship)

b). determine the normal form for each entity, if it is not in 3NF, convert it to 3NF.



ANSWER:

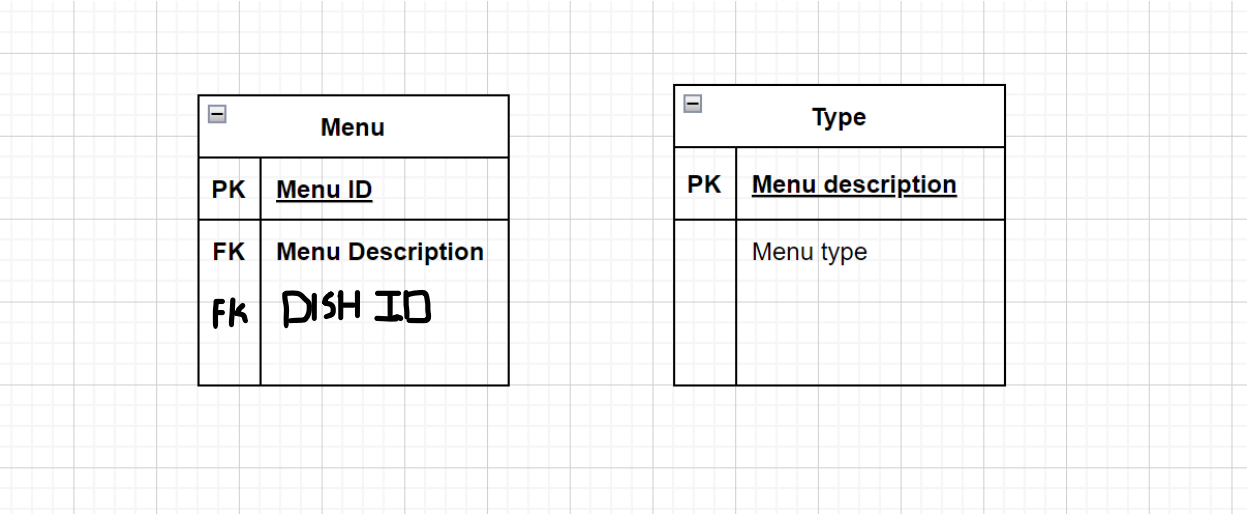
MENU:

- it's in 1NF (because there are no multi-valued attributes)
- It's in 2NF (because there is no partial dependency)
- But it's not in 3NF because there is transitive dependency where Menu Type (Y) is dependent on Menu Description(B) and Menu Description(B) is dependent on Menu ID(X).

Solution:

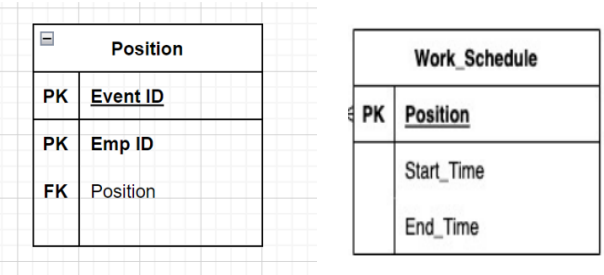
- So, create a new table called Type with Menu Description (B) and Menu Type (Y)
- Set Menu Description (B) as PK in new table and as a FK in old table.(Menu table)
- Remove Menu Type (Y) from the old table. (Menu Table)

Hence 3NF Design for Menu table is as follows:



EVENT: it is already in 3NF.

WORK SCHEDULE: it is not in 3NF. Because of transitive dependency. So the below table is introduced.



DISH: its not in 1NF, 2NF, and in 3NF

- The ingredient is a multivalued attribute. So, it's not 1NF.

  Solution:
  - Remove multi-values by adding extra rows.

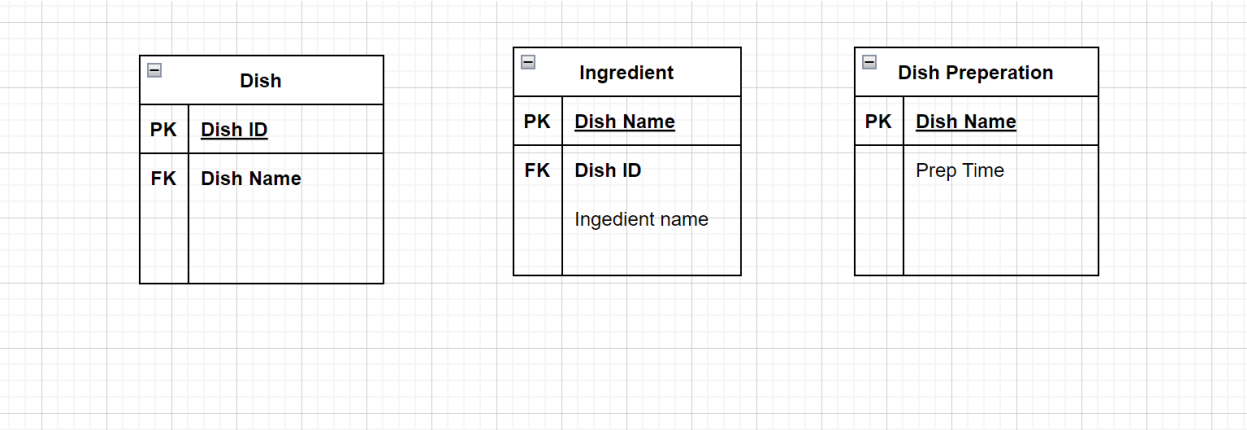
- There is Transitive dependency where, Ingredient(Y) depends on Dish Name(B) and Dish Name(B) is dependent on Dish ID(X). So, it's not in 3NF.

  Solution:
  - Create new table called INGREDIENT with Dish Name(B) and Ingredient(Y)
  - Set Dish Name(B) as PK in new table and as a FK in old table.
  - Remove Ingredient(Y) from the old table.

- There is partial dependency where prep Time(Y) is partially dependent on Dish Name.(X) So, it's not in 2NF.

  Solution:
  - Create a new table called Dish Preparation with Dish Name(X) and Prep Time(Y)
  - Set Dish Name as PK in new table and as a FK in old table.
  - Remove Prep Time (Y) from the old table.

Hence 3NF Design for Dish table is as follows:



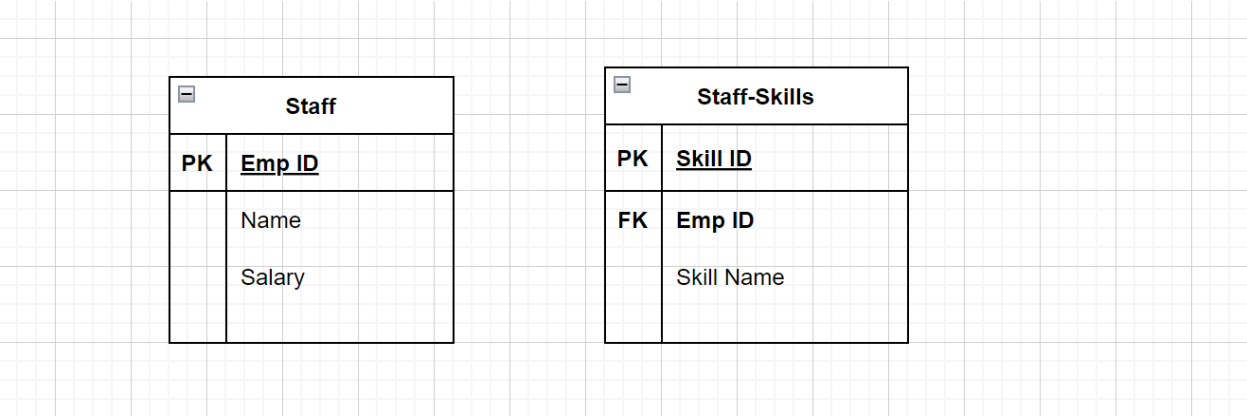
STAFF:

- it's not in 1NF, because The Skills is a multivalued attribute.

  Solution:
  - Remove multi-values by adding extra rows. (or)
  - Create a separate table called (staff-skills table) with attributes Skill ID, Skill Name and Emp ID.

- assigning Skill ID as a PK and Emp ID as a FK in the new table (staff-skills table).
- Remove skills column in the original table. (Staff table)

Hence New Design for Staff table is as follows:

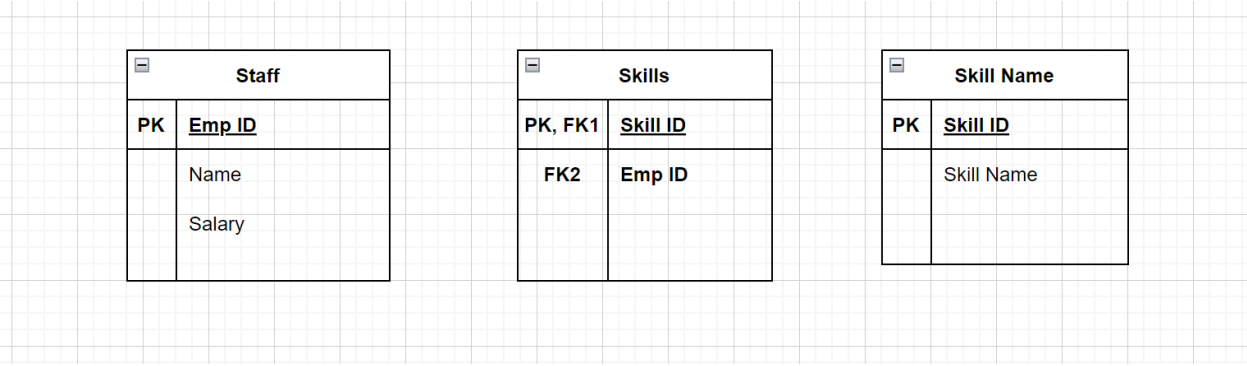


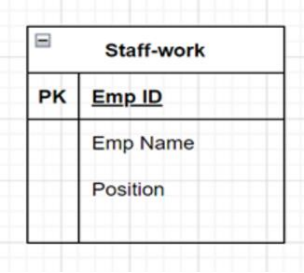
- The Skills table is not in 2NF because there is a partial dependency.
- Because if Skills ID + Emp ID together forms as a candidate key, Skill name(Y) is partially dependent on Skill ID(X) Which is a part of Candidate key.

Solution:

- So crate a new table called skills with Skill ID(X) and Skill name(Y)
- Set Skill ID as a PK in new table(skill table) and as a FK in old table.(staff-skills table)
- Remove Skill Name(Y) from the old table.(staff-skills table)

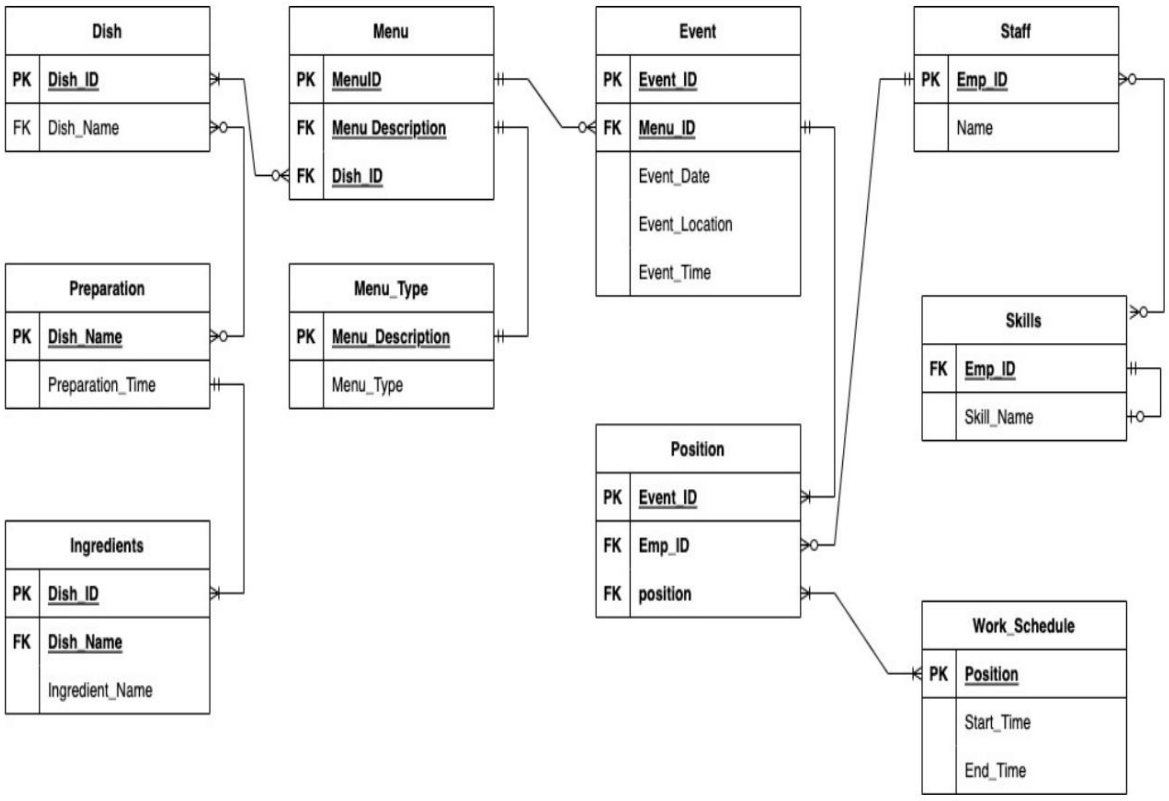
Hence New Design for Staff table is as follows:





- Staff -work table is introduced to make sure each staffer is assigned a role or to check position of each staff. Because there is also a supervisor who is also a staff.

The final 3NF ERD is as follows:



Q4. DB Design [50 points]

System Descriptions

- Build a DB for hotel bookings
- Each hotel only has two types of rooms: 1-double bedroom, and two-twin bedroom
- Each hotel may have multiple rooms
- Clients can place orders. In one order, they can book multiple rooms from a same hotel

Draw ERD to design a DB in 3NF

Note: you can directly give the ERD in physical DB design, but you must make sure it is in 3NF.

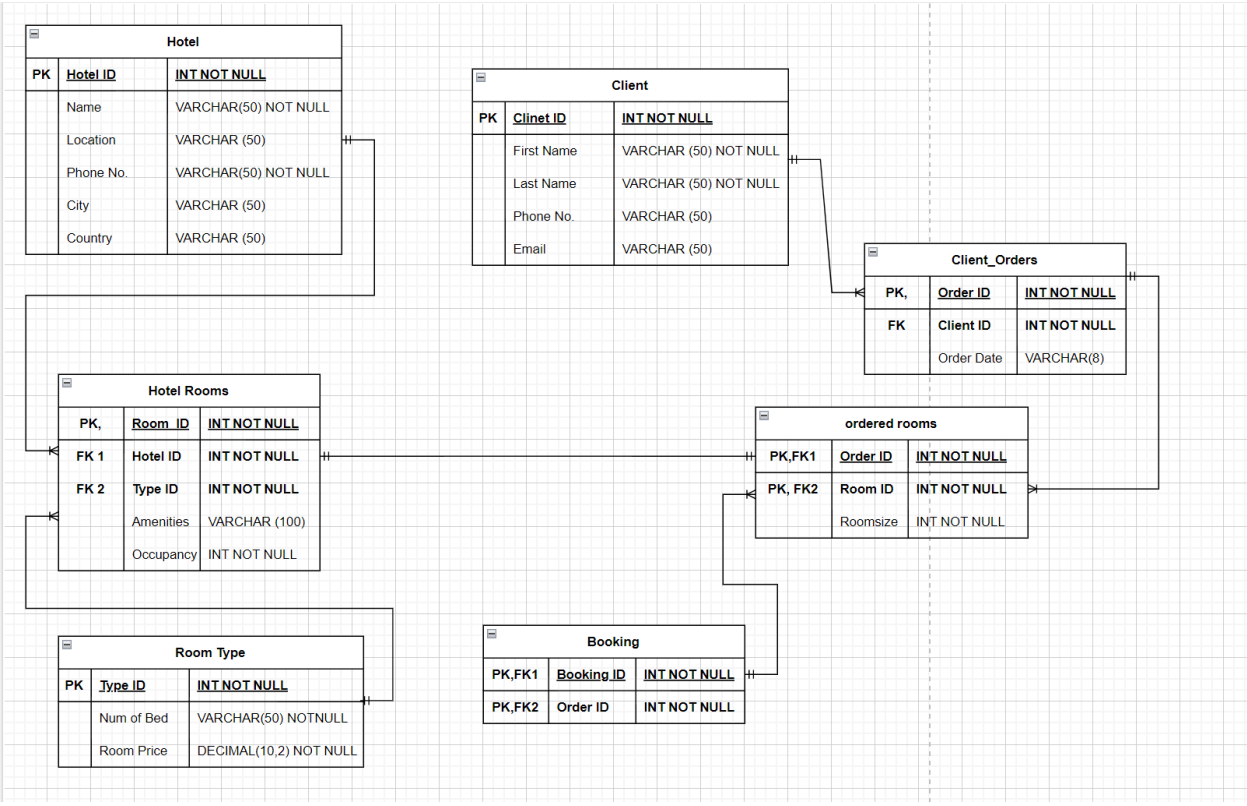
ANSWER:

The Main entities we have with attributes and PK's:

1) **Hotel** (HotelID(pk), Name, Location, PhNo.,City,Country)
2) **Client** (Client ID(pk), F_Name, L_Name, PhNo., Email)

Additional entities that need to be added in the process of 3NF Normalization:

3) **Hotel_Rooms** (Room ID-pk&fk , Hotel ID – fk , Type ID – fk ,Amenities, Occupancy)
4) **Room Type** (Type ID – pk, Num of Bed, Room price)
5) **Client_Orders** (Order ID -pk, Client ID – pk, Order_date)
6) **Ordered_Rooms** (Oder ID – pk&fk, Room ID – pk&fk, Room size)
7) **Booking** (Booking ID – pk&fk , Oder ID – pk&fk)



FEEDBACK & SCORE : 99

HW3

Q4. -1 Relationship between hotel rooms and ordered rooms must be 1:M not 1:1

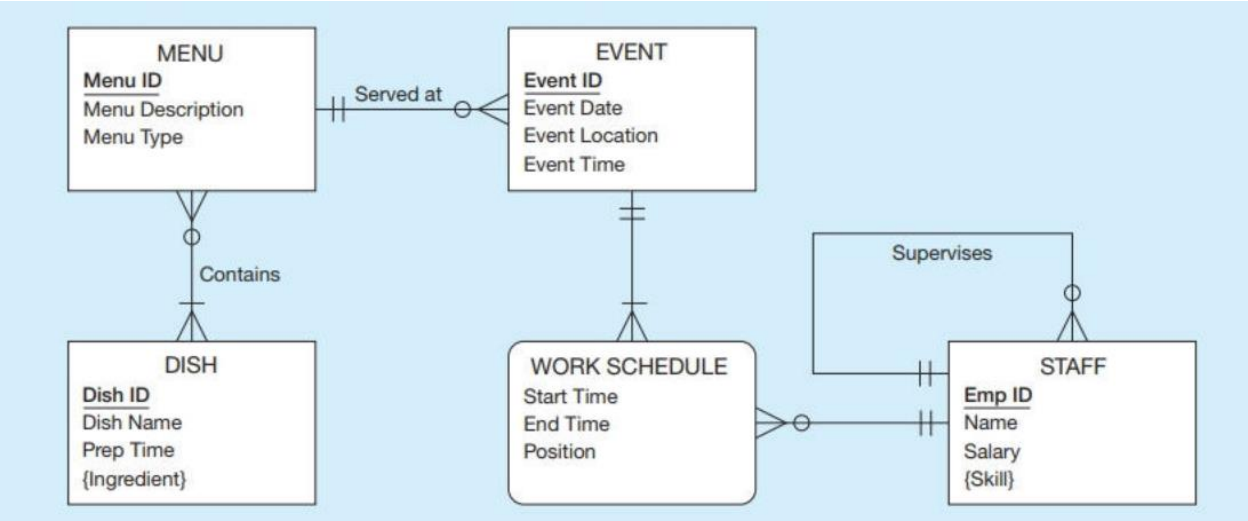
Homework 4

Your Name: Naga Satya Silpa Annadevara

Student ID: A20517818

Undergraduate (Yes/No): NO

Q1. Adding data types to the attributes in the figure below. Note, you do not need to draw ERD, you can simply give a list with table name and attribute names, along with the data types in Oracle. [10]



ANSWER:

MENU:

ATTRIBUTE NAME	DATA TYPE
Menu ID	INT or NUMBER **(Not Null)**
Menu Description	VARCHAR2 (100)
Menu Type	VARCHAR2 (50)

EVENT:

ATTRIBUTE NAME	DATA TYPE
Event ID	INT or NUMBER (Not Null)
Event Date	DATE
Event Location	VARCHAR2 (100)
Event Time	TIMESTAMP

DISH:

ATTRIBUTE NAME	DATA TYPE
Dish ID	INT or NUMBER (Not Null)
Dish Name	VARCHAR2 (50)
Prep Time	TIMESTAMP
Ingredient	VARCHAR2 (50)

WORK SCHEDULE:

ATTRIBUTE NAME	DATA TYPE
Start Time	TIMESTAMP
End Time	TIMESTAMP
Position	VARCHAR2 (20)

STAFF:

ATTRIBUTE NAME	DATA TYPE
Emp ID	NUMBER (Not Null)
Emp Name	VARCHAR2 (40)
Salary	DECIMAL or FLOAT or NUMBER
Skill	VARCHAR2 (50)

Q2. Below is the design for the case study of hotel-rooms-booking discussed in the class. [40]

System/Application Requirements

- We want to build hotel room booking system.
- Assume there are several hotels in Chicago.
- Each hotel only has two types of the rooms – Standard Room with 2 beds, Standard Room with 1 big bed
- Customer can book rooms from our website. But the availability may vary from hotels to hotels

Our design:

- Hotel (HID, Name, Address, Cell, …)
- RoomType (TypeID, NumBeds, RoomSize, …)
- HotelRooms (RoomID, HID, TypeID, Quantity)
- Customer (CID, Name, Address, Email, …)
- Order (OrderID, CID, CheckinDate, CheckoutDate)
- RoomsInOrder (OrderID, RoomID, RoomQuantity)

a). Identify the PK and FK in each entity [10]

ANSWER:

1. Hotel (HID, Name, Address, Cell, …)

- Primary key: HID
- Foreign key: None

2. Room Type (TypeID, NumBeds, RoomSize, …)

- Primary key: Type ID
- Foreign key: None

3. HotelRooms (RoomID, HID, TypeID, Quantity)

- Primary key: Room ID
- Foreign key: HID, Type ID

4. Customer (CID, Name, Address, Email, …)

- Primary key: CID
- Foreign key: None

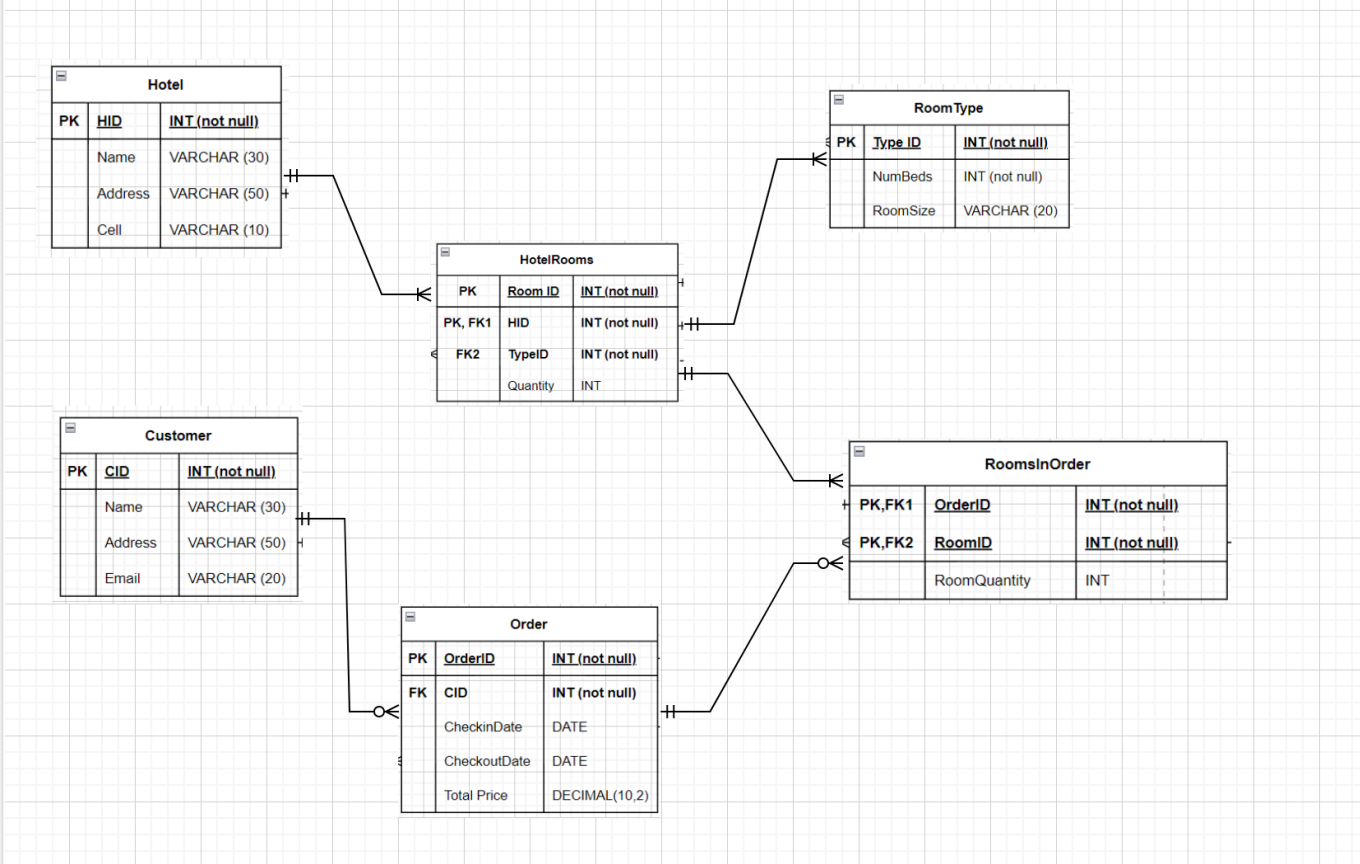
5. Order (OrderID, CID, CheckinDate, CheckoutDate)

- Primary key: Order ID
- Foreign key: CID

6. RoomsInOrder (OrderID, RoomID, RoomQuantity)

- Primary key: Order ID + Room ID
  Foreign key: Order ID + Room ID

b). Draw ERD for physical DB design [10]



NOTE: The ID's Can use the datatype NUMBER as well

c). Now, we want to add "price". Note that there would be unit price for the rooms, as well as the price in the orders. Special notes: hotels may release difference prices for different time periods. Complete your design in BCNF by showing your physical design by ERD [20]

ANSWER:

As the price is introduced into the design, we have to add 3 more NEW entities with attributes as follows:

- HotelRoom Price (PriceID(pk) , RoomID(pk&fk) , Unit Price)
- Ordered Price (RoomsInOrderID(pk,fk) ,PriceID(pk,fk), Total Price)
- Periodical Price (PeriodID (pk) ,TypeID (fk), RoomID(fk), Current Price)

  (As mentioned in the question that hotels may release difference prices for different time periods)

In addition to the above entities, we need to add attributes to the old entities according to the question as follows:

- adding TotalPrice in Order Entity.

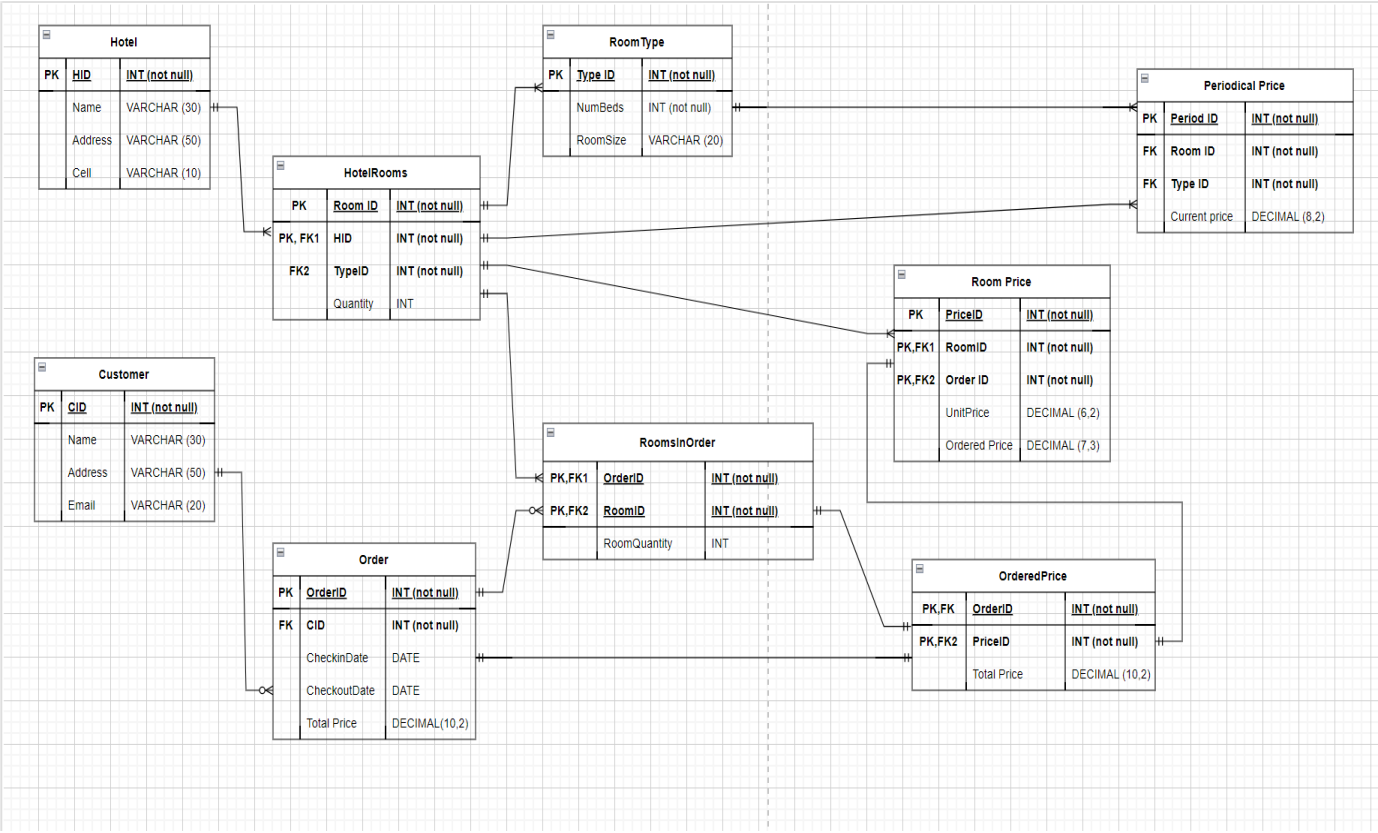
- Unit Price to the HotelRooms entity (this attribute will be moved to the new entity called HotelroomsPrice after normalization as shown in the figure below)

Hence, The final list of entities we have after adding price to the design and after normalization to BCNF are :

1. Hotel
2. HotelRooms
3. Room Type
4. Customer
5. Order
6. RoomsInOrder
7. HotelRoom Price
8. Ordered Price

   Periodical Price (As mentioned in the question that hotels may release difference prices for different time periods)

BCNF in Physical design by ERD:

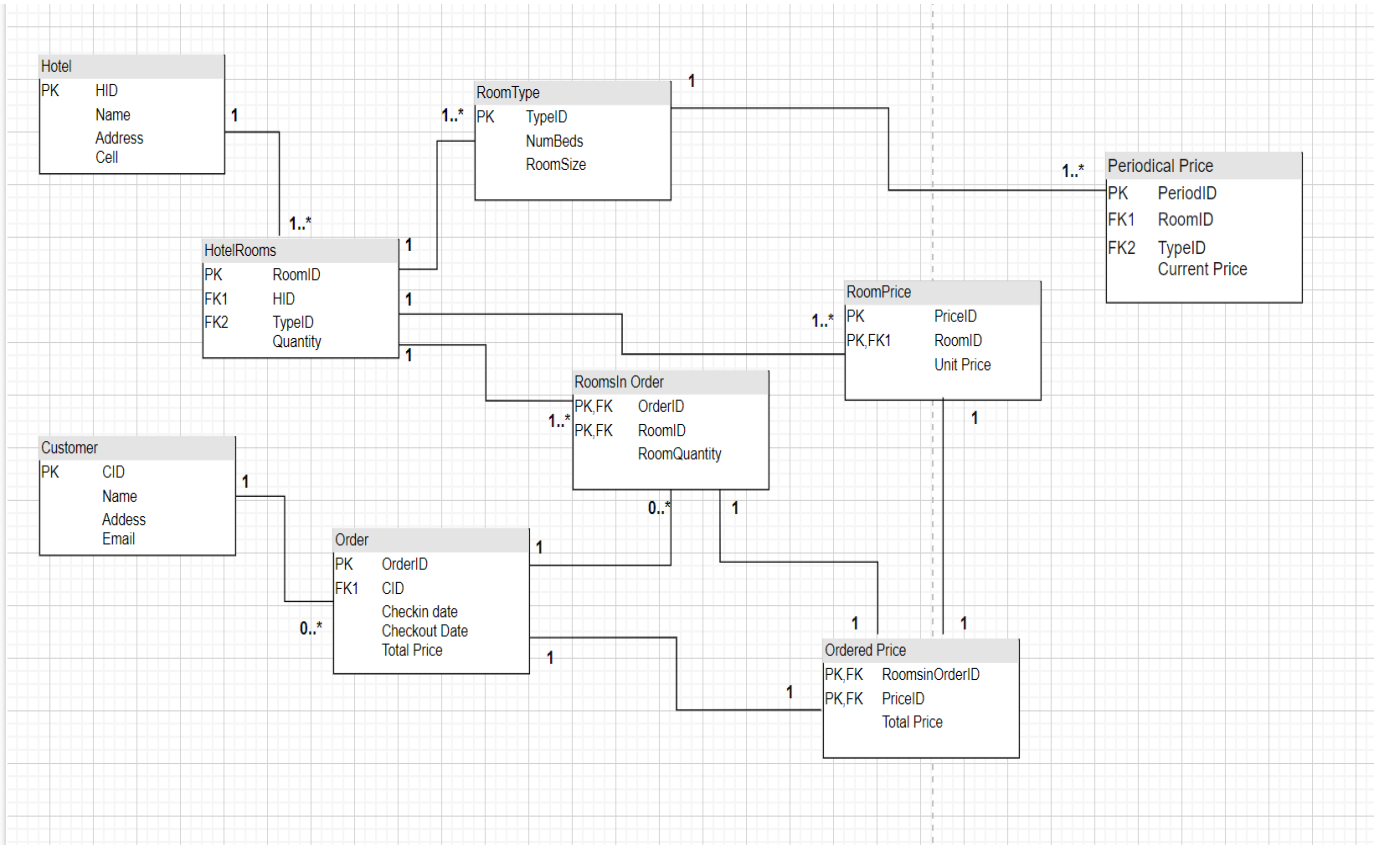


NOTE: The ID's Can use the datatype NUMBER as well

Q3. [10] Draw the final design in Q3 by using UML.

ANSWER:

Final design in UML:



Q4. [40] Using DDL and DML for the following.

Download the W5_MovieRatings.zip file, unzip it. You will find four csv files, each of which is an entity/table. Therefore, there are 4 tables: Users, Movies, Ratings, Tags.

[4] Identify PK and FK in these tables. Use * to indicate PK, and ___ to tell FK. For example, Student (StudentID*, DeptID, Firstname, Lastname)

[8] Use create statements to create these tables in Oracle, show your SQL coding

[8] Use insert statements to insert values in these tables, show your SQL coding

[4] Add a column “TagDate” in the table Tags. After that, insert date values for the current entries in table Tags. Note, you can use any date values, show your SQL coding.

[8] Drop table movies, show your SQL coding.

[4] In table Ratings, change the data type of ratings to Number (2, 1) , show your SQL coding

[4] Assume that users can only give ratings, like, 1, 2, 3, 4, 5. In table Ratings, add a check constraint to guarantee that the input values for the column rating must be a value from (1, 2, 3, 4, 5)

ANSWER:

1. Users Table (UserID, Email, BirthYear, City)

- Primary key: UserID
- Foreign key: None

2. Movies Table (MovieID, Title)

- Primary key: Movie ID
- Foreign key: None

3. Ratings (UserID, MovieID, Rating)

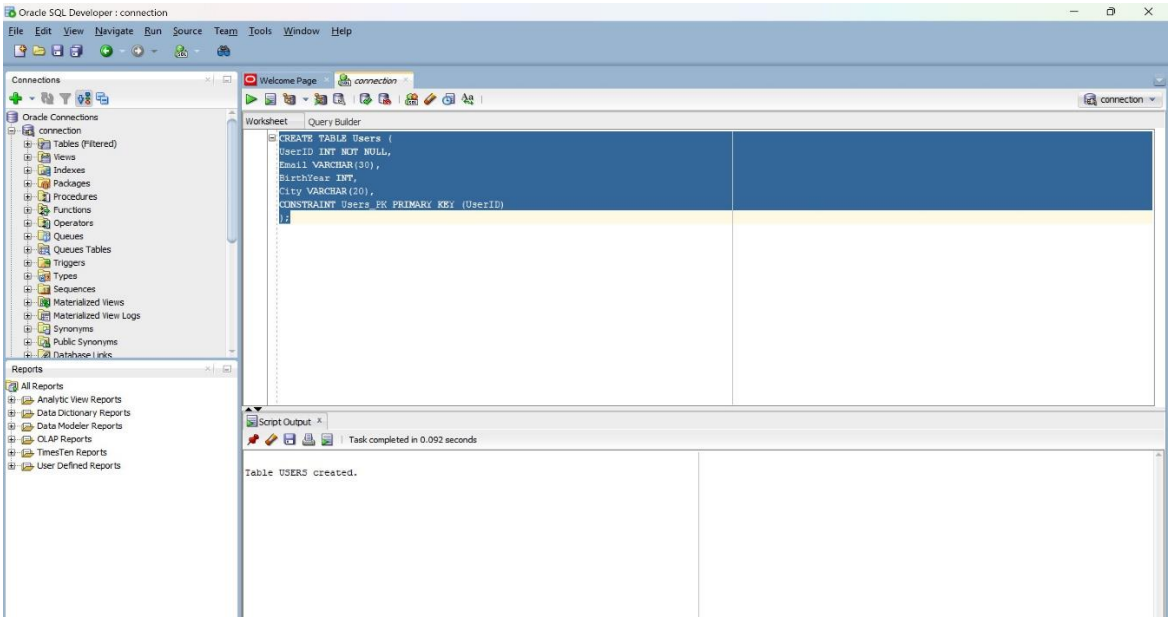
- Primary key: ratings + Movie ID +UserID
- Foreign key: UserID, Movie ID

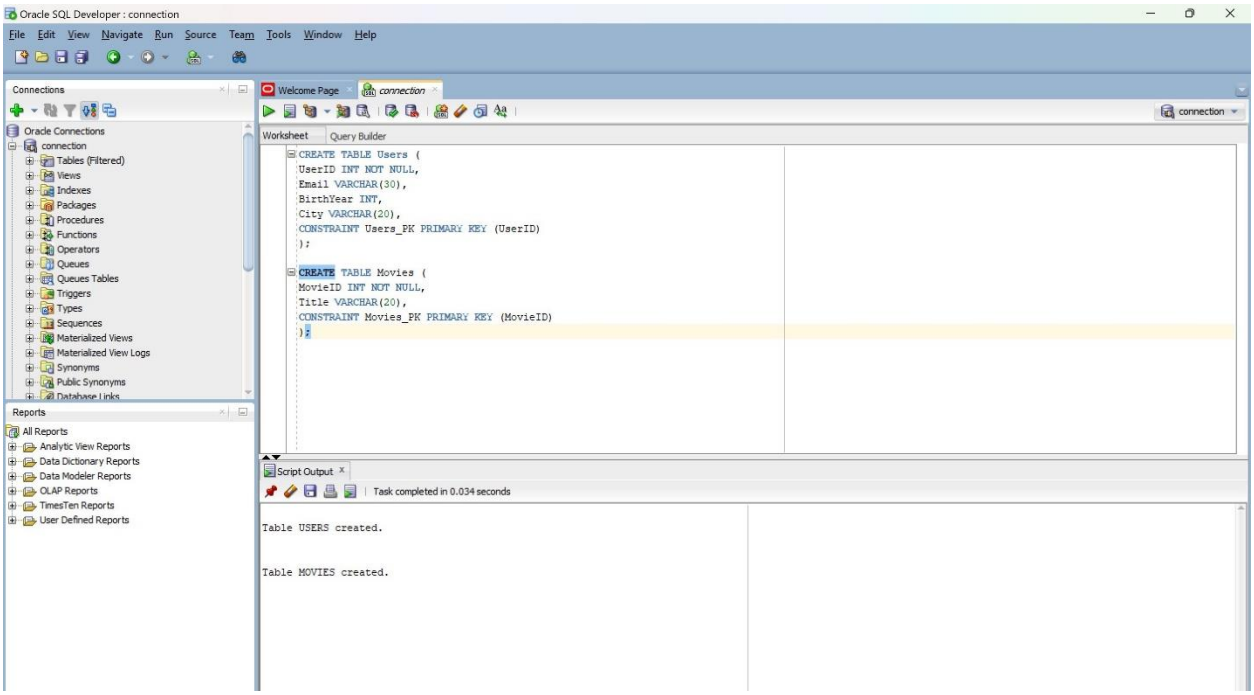
4. Tags (UserID, MovieID, Tag)

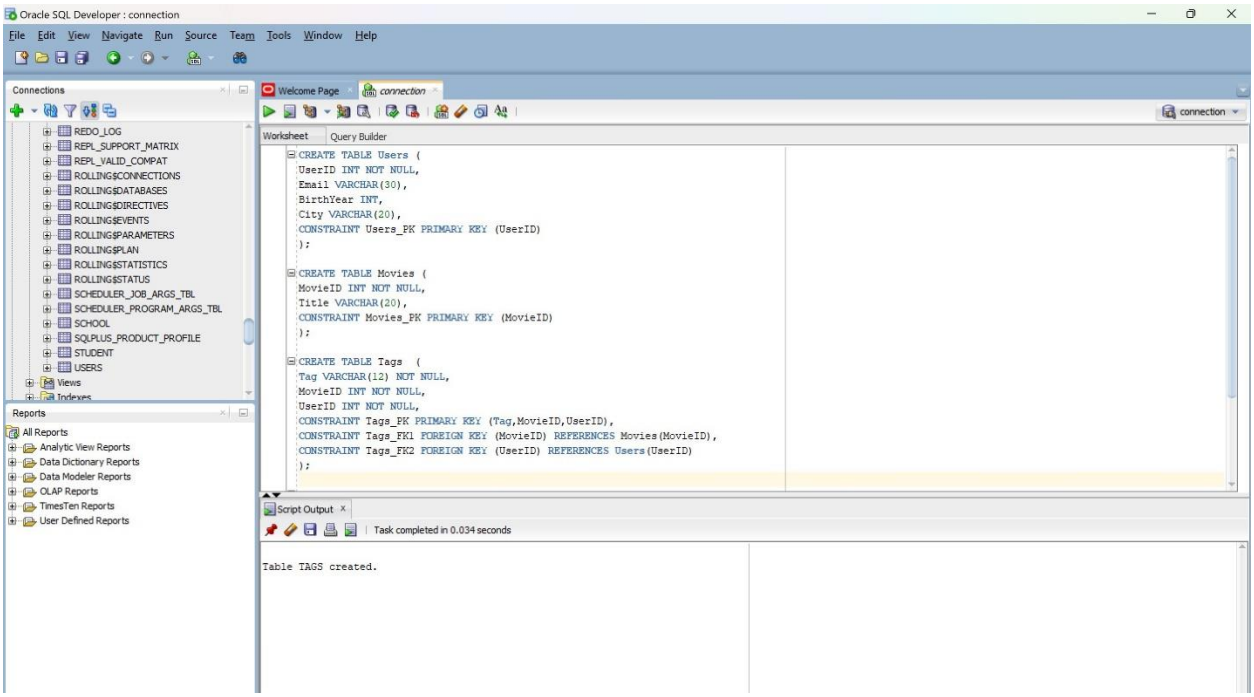
- Primary key: Tag + MovieID + UserID
- Foreign key: UserID, Movie ID

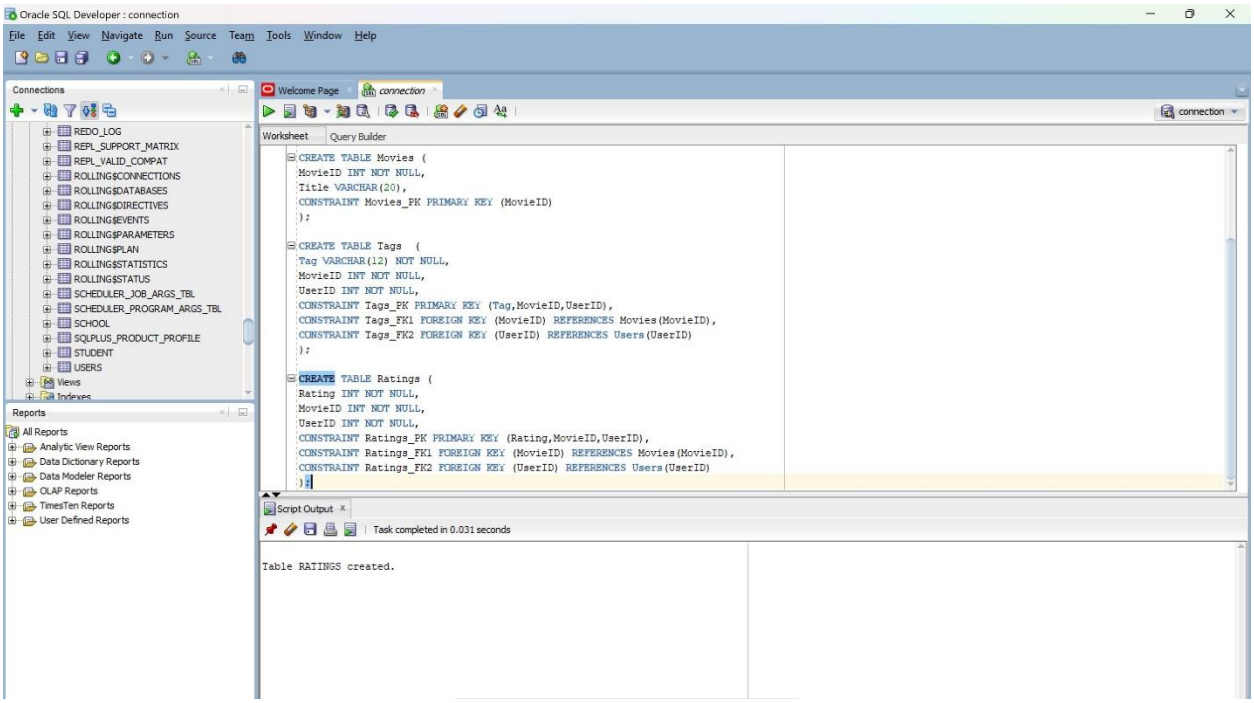
Oracle ScreenShots (SQL Coding):

Creating Tables (Users, Movies, Ratings, Tags) :

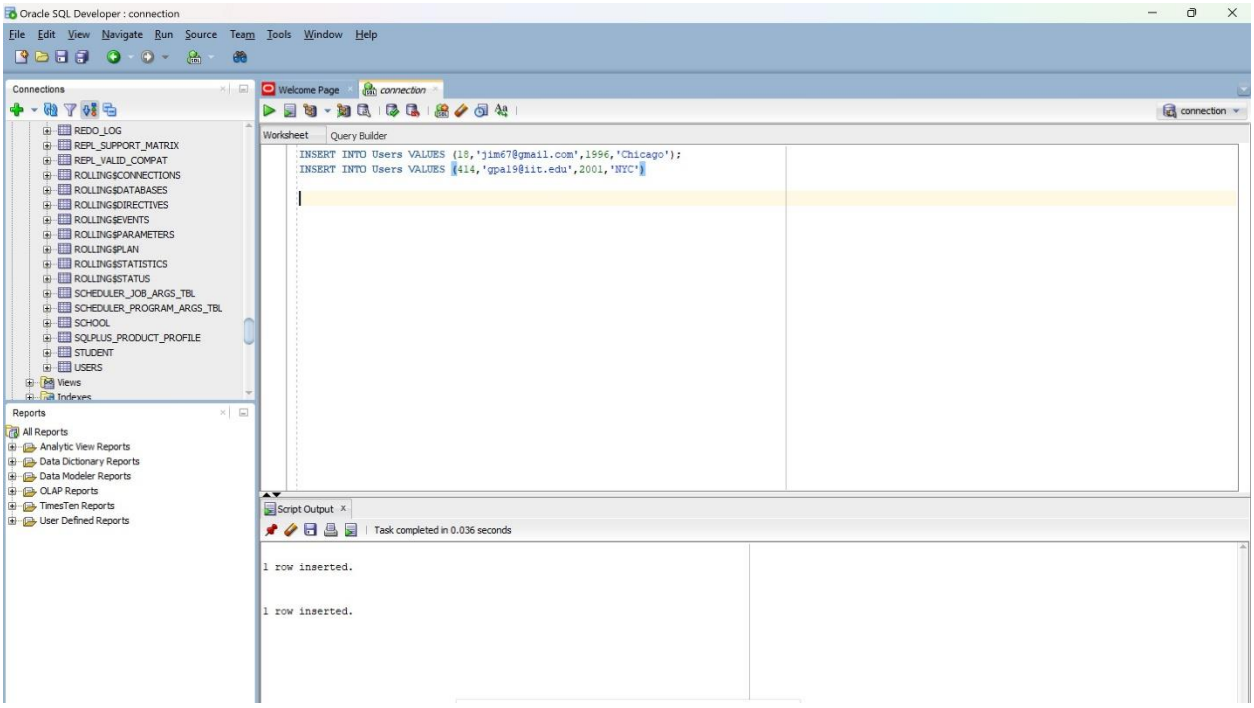


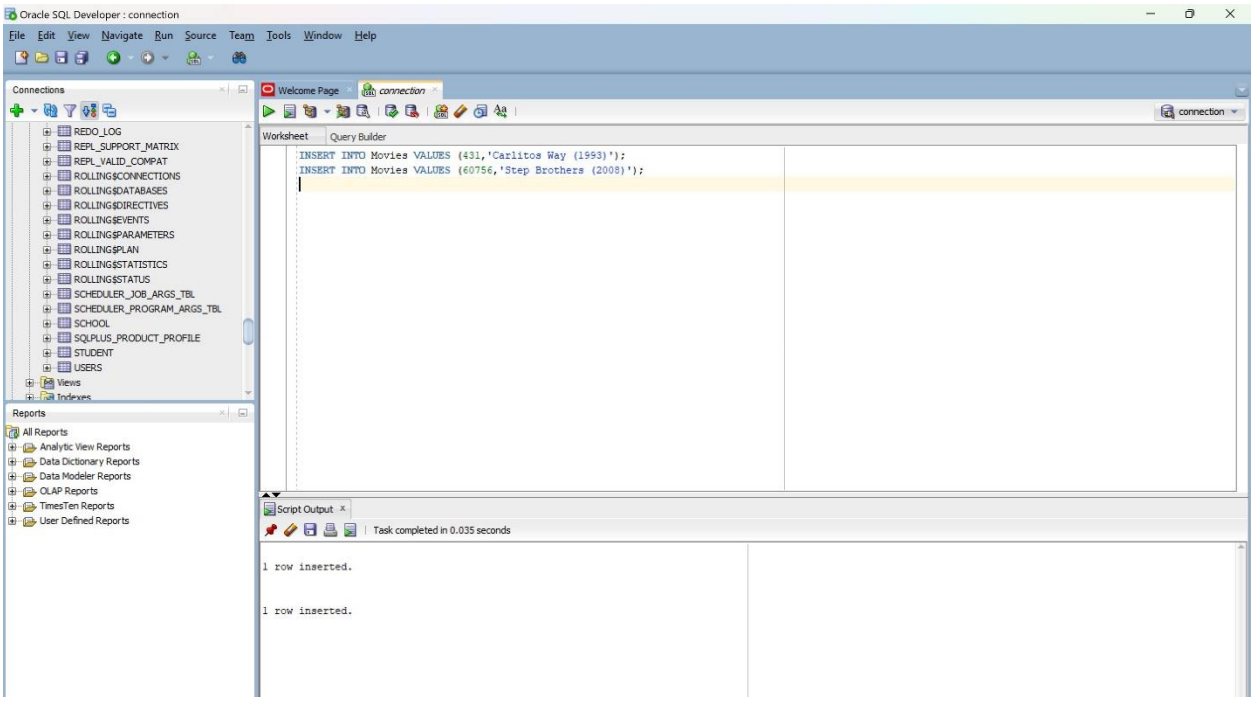

Oracle SQL Developer : connection
File Edit View Navigate Run Source Team Tools Window Help
Connections
Oracle Connections
connection
Tables (Filtered)
Views
Indexes
Packages
Procedures
Functions
Operators
Queues
Queues Tables
Triggers
Types
Sequences
Materialized Views
Materialized View Logs
Synonyms
Public Synonyms
Database Links
Reports
All Reports
Analytic View Reports
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
TimesTen Reports
User Defined Reports
Welcome Page
connection
Worksheet
Query Builder
CREATE TABLE Users (
UserID INT NOT NULL,
Email VARCHAR(30),
BirthYear INT,
City VARCHAR(20),
CONSTRAINT Users_PK PRIMARY KEY (UserID)
);
CREATE TABLE Movies (
MovieID INT NOT NULL,
Title VARCHAR(20),
CONSTRAINT Movies_PK PRIMARY KEY (MovieID)
);
Script Output
Task completed in 0.034 seconds
Table USERS created.
Table MOVIES created.

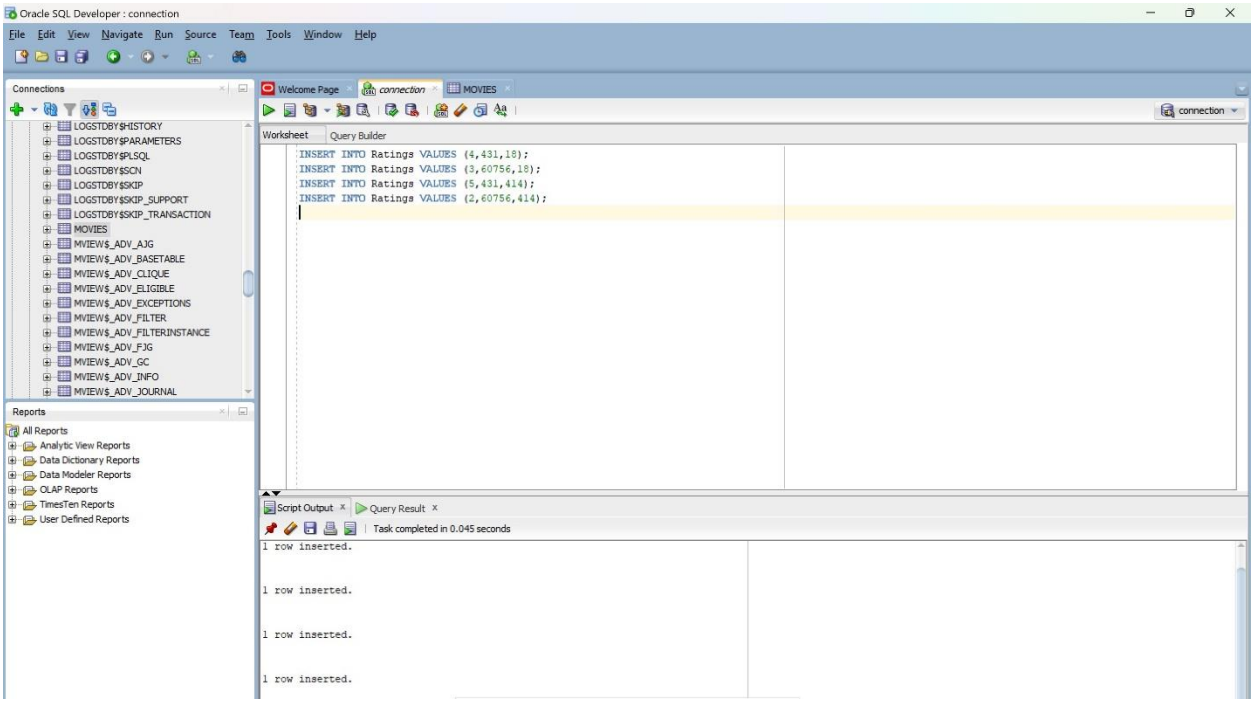

Oracle SQL Developer : connection
File Edit View Navigate Run Source Team Tools Window Help
Connections
REDO_LOG
REPL_SUPPORT_MATRIX
REPL_VALID_COMPAT
ROLLING$CONNECTIONS
ROLLING$DATABASES
ROLLING$DIRECTIVES
ROLLING$EVENTS
ROLLING$PARAMETERS
ROLLING$PLAN
ROLLING$STATISTICS
ROLLING$STATUS
SCHEDULER_JOB_ARGS_TBL
SCHEDULER_PROGRAM_ARGS_TBL
SCHOOL
SQLPLUS_PRODUCT_PROFILE
STUDENT
USERS
Views
Indexes
Reports
All Reports
Analytic View Reports
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
TimesTen Reports
User Defined Reports
Welcome Page
connection
Worksheet
Query Builder
CREATE TABLE Users (
UserID INT NOT NULL,
Email VARCHAR(30),
BirthYear INT,
City VARCHAR(20),
CONSTRAINT Users_PK PRIMARY KEY (UserID)
);
CREATE TABLE Movies (
MovieID INT NOT NULL,
Title VARCHAR(20),
CONSTRAINT Movies_PK PRIMARY KEY (MovieID)
);
CREATE TABLE Tags (
Tag VARCHAR(12) NOT NULL,
MovieID INT NOT NULL,
UserID INT NOT NULL,
CONSTRAINT Tags_PK PRIMARY KEY (Tag,MovieID,UserID),
CONSTRAINT Tags_FK1 FOREIGN KEY (MovieID) REFERENCES Movies(MovieID),
CONSTRAINT Tags_FK2 FOREIGN KEY (UserID) REFERENCES Users(UserID)
);
Script Output
Task completed in 0.034 seconds
Table TAGS created.

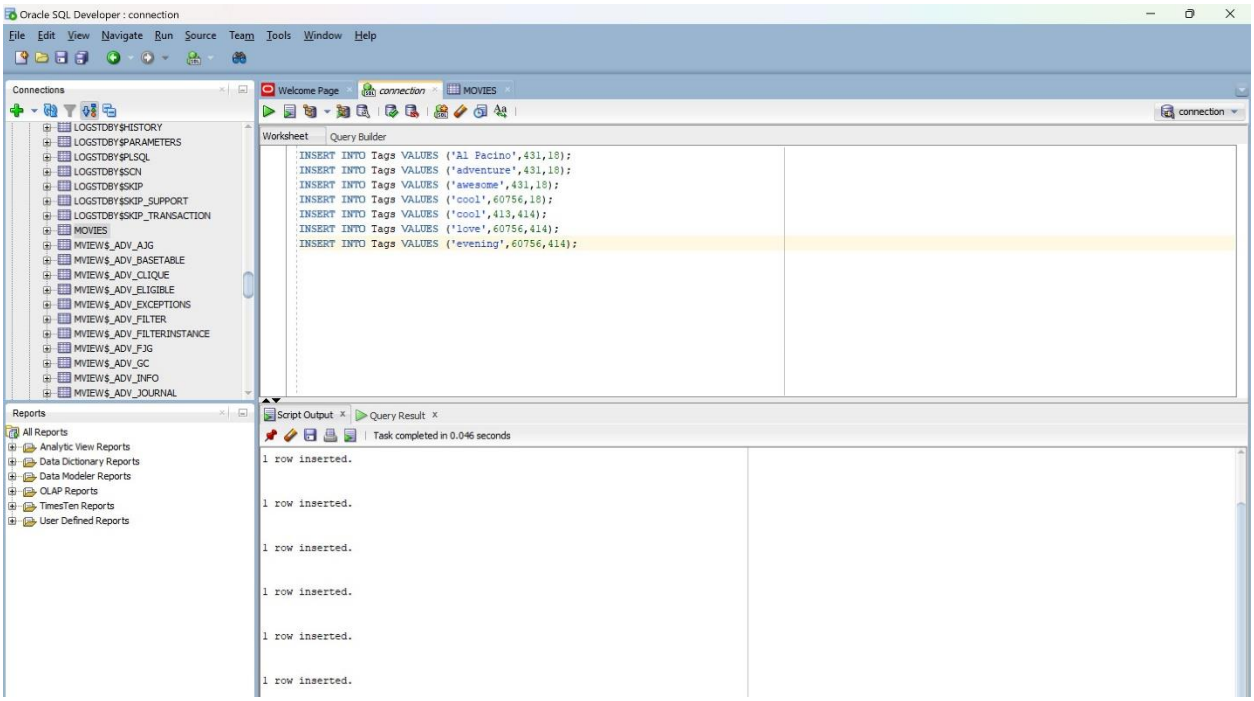


Inserting Values into the Tables:

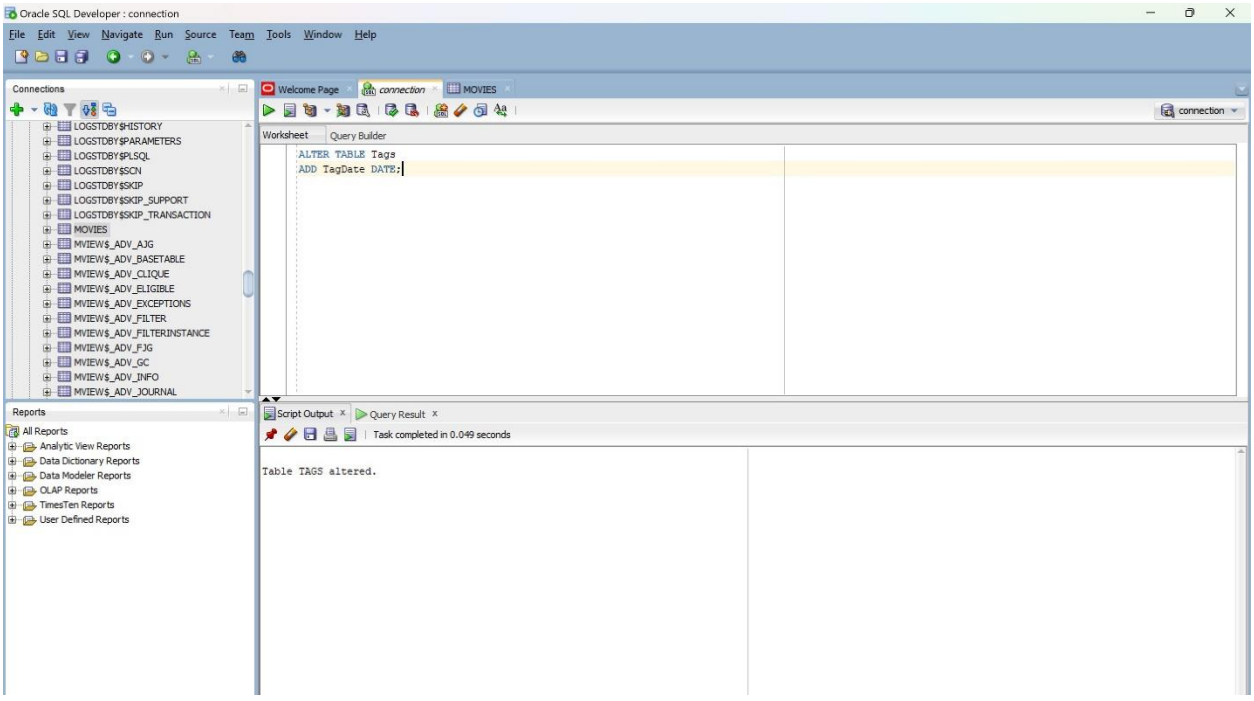


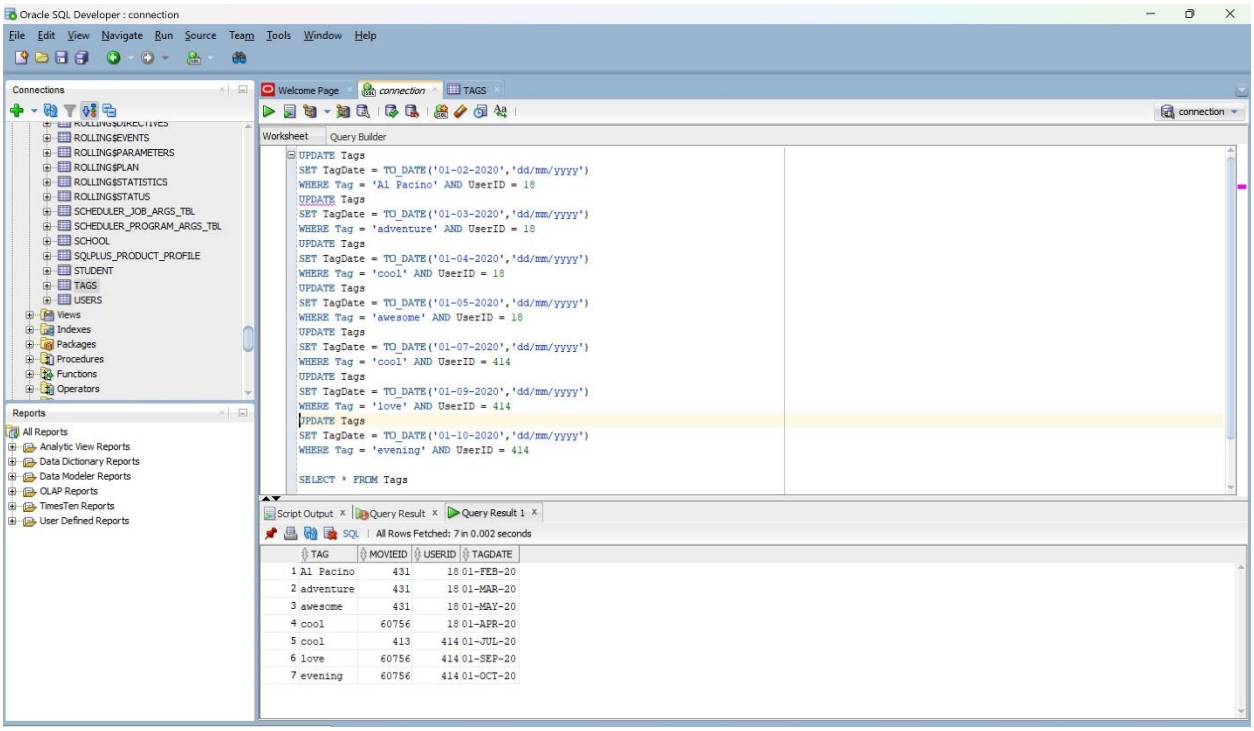

Oracle SQL Developer : connection
File Edit View Navigate Run Source Team Tools Window Help
Connections
REDO_LOG
REPL_SUPPORT_MATRIX
REPL_VALID_COMPAT
ROLLING$CONNECTIONS
ROLLING$DATABASES
ROLLING$DIRECTIVES
ROLLING$EVENTS
ROLLING$PARAMETERS
ROLLING$PLAN
ROLLING$STATISTICS
ROLLING$STATUS
SCHEDULER_JOB_ARGS_TBL
SCHEDULER_PROGRAM_ARGS_TBL
SCHOOL
SQLPLUS_PRODUCT_PROFILE
STUDENT
USERS
Views
Indexes
Reports
All Reports
Analytic View Reports
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
TimesTen Reports
User Defined Reports
Welcome Page
connection
connection
Worksheet
Query Builder
INSERT INTO Movies VALUES (431,'Carlitos Way (1993)');
INSERT INTO Movies VALUES (60756,'Step Brothers (2008)');
Script Output
Task completed in 0.035 seconds
1 row inserted.
1 row inserted.


Oracle SQL Developer : connection
File Edit View Navigate Run Source Team Tools Window Help
Connections
LOGSTDBY$HISTORY
LOGSTDBY$PARAMETERS
LOGSTDBY$PLSQL
LOGSTDBY$SCN
LOGSTDBY$SKIP
LOGSTDBY$SKIP_SUPPORT
LOGSTDBY$SKIP_TRANSACTION
MOVIES
MVIEW$_ADV_AJG
MVIEW$_ADV_BASETABLE
MVIEW$_ADV_CLIQUE
MVIEW$_ADV_ELIGIBLE
MVIEW$_ADV_EXCEPTIONS
MVIEW$_ADV_FILTER
MVIEW$_ADV_FILTERINSTANCE
MVIEW$_ADV_FJG
MVIEW$_ADV_GC
MVIEW$_ADV_INFO
MVIEW$_ADV_JOURNAL
Reports
All Reports
Analytic View Reports
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
TimesTen Reports
User Defined Reports
Welcome Page
connection
MOVIES
connection
Worksheet
Query Builder
INSERT INTO Ratings VALUES (4,431,18);
INSERT INTO Ratings VALUES (3,60756,18);
INSERT INTO Ratings VALUES (5,431,414);
INSERT INTO Ratings VALUES (2,60756,414);
Script Output
Query Result
Task completed in 0.045 seconds
1 row inserted.
1 row inserted.
1 row inserted.
1 row inserted.

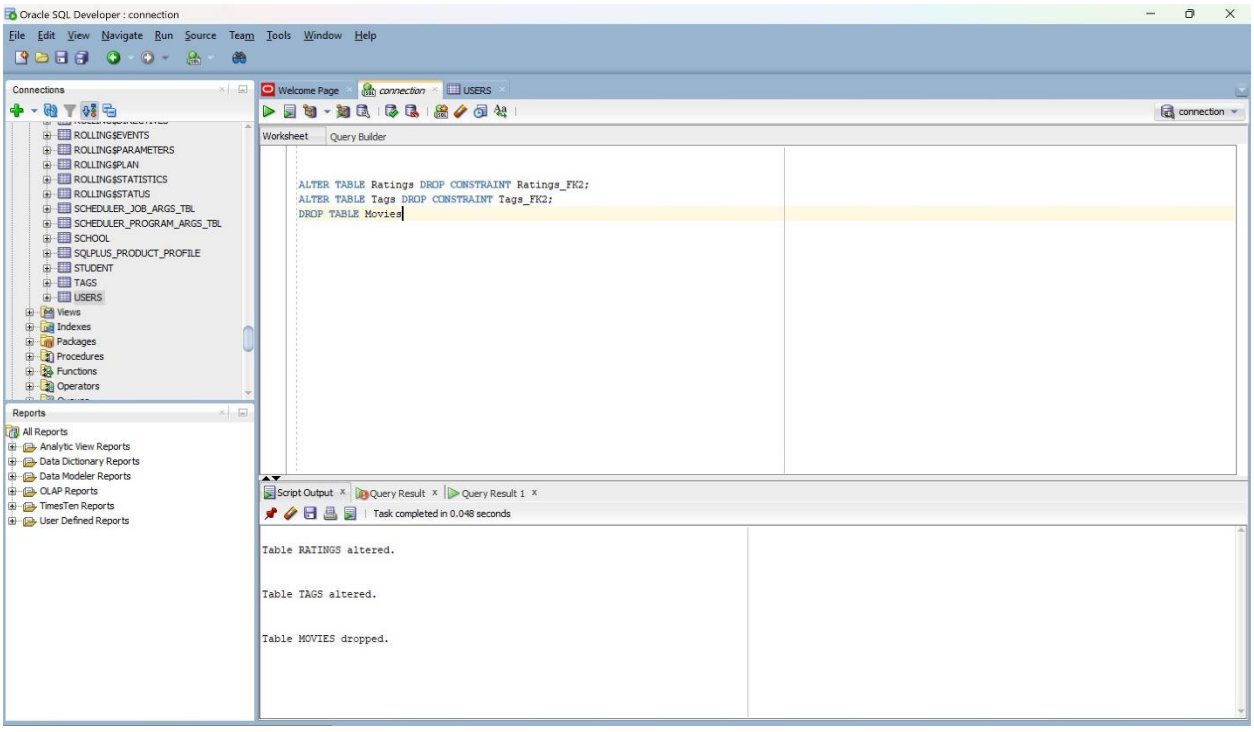


Add a column “TagDate” in the table Tags. After that, insert date values for the current entries in table Tags. Note, you can use any date values, show your SQL coding.

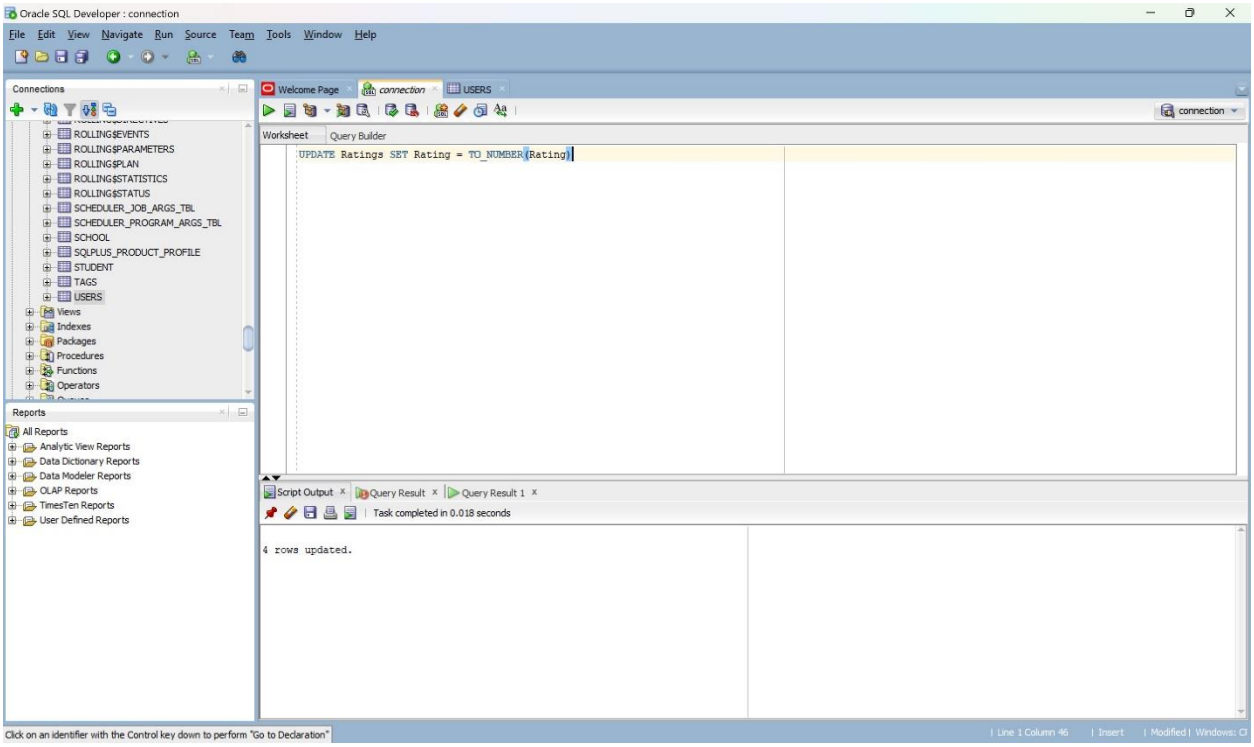


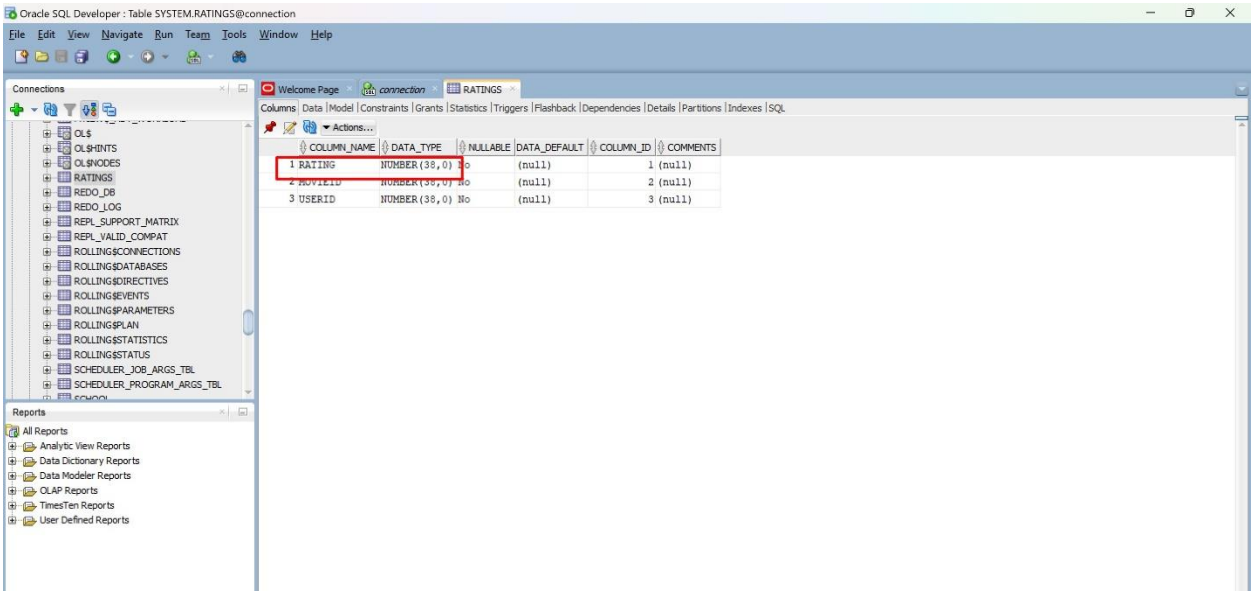


Drop table movies, show your SQL coding.

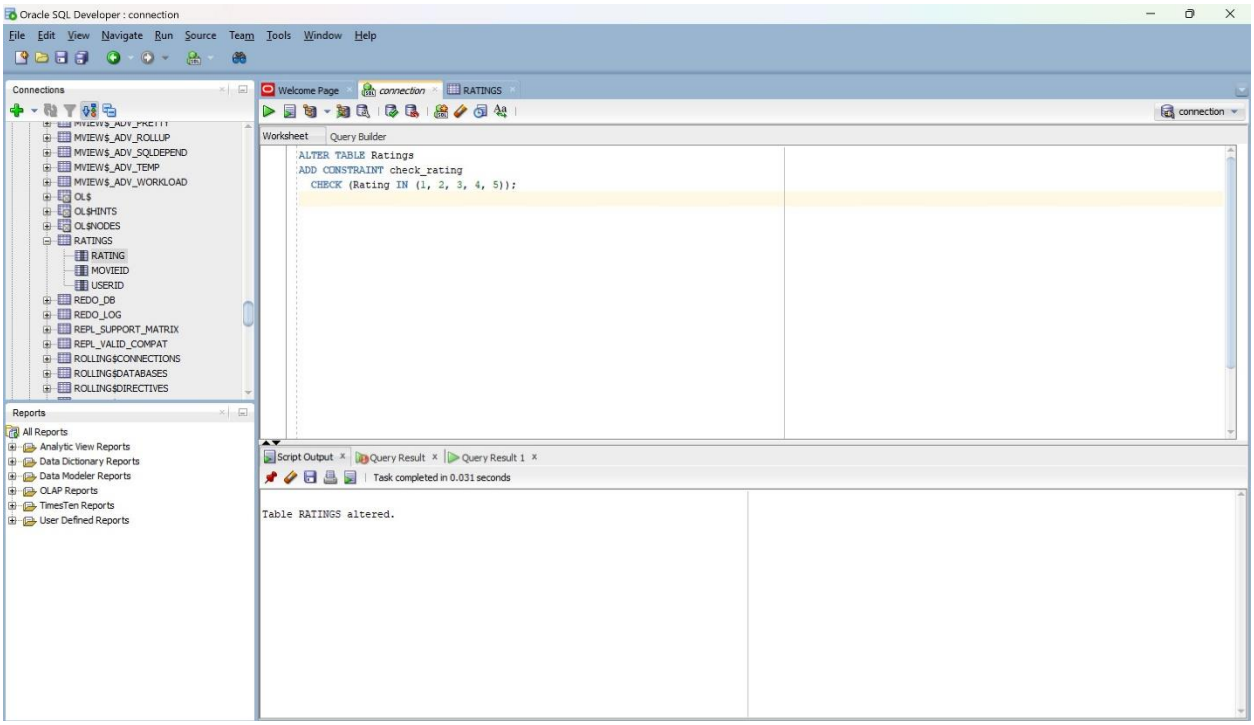


In table Ratings, change the data type of ratings to Number (2, 1) , show your SQL coding:





Assume that users can only give ratings, like, 1, 2, 3, 4, 5. In table Ratings, add a check constraint to guarantee that the input values for the column rating must be a value from (1, 2, 3, 4, 5)



FEEDBACK AND GRADE:

I got 100 **********

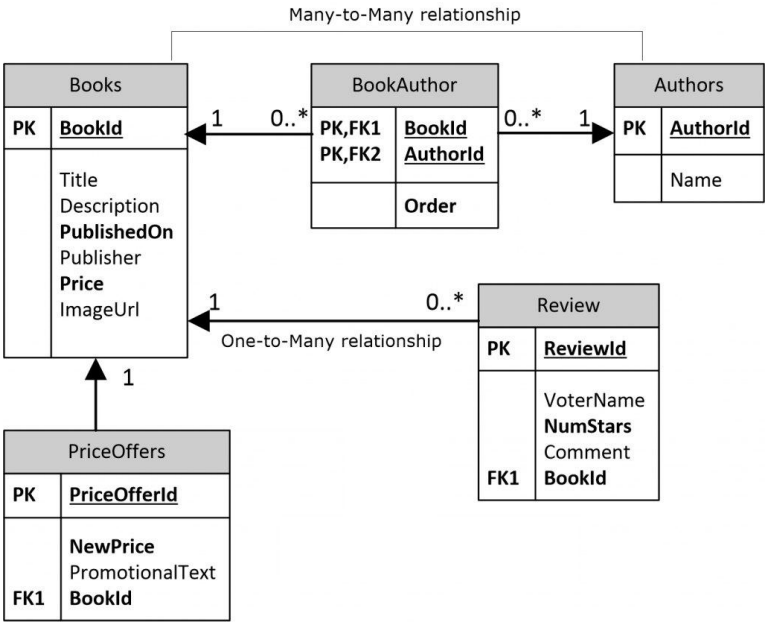
Homework 5

Your Name: Naga Satya Silpa Annadevara

Student ID: A20517818

---

1. Use DDL to answer the following questions. Note: you do not need to run SQL in Oracle. Just give the answers for the following requests [40]



1). Use DDL to create a database named as "BookInfo". Create the tables, except the table "BookAuthor" [5]

ANSWER:

```
CREATE DATABASE BookInfo;

CREATE TABLE Books (
BookID INT NOT NULL,
Description VARCHAR2(100),
Title VARCHAR2(30),
PublishedOn DATE,
Publisher VARCHAR2(20),
Price FLOAT,
ImageUrl VARCHAR2(100)
CONSTRAINT Books_PK PRIMARY KEY (BookID)
);
```

```
CREATE TABLE PriceOffers (
PriceOfferID INT NOT NULL,
NewPrice DECIMAL(6,2),
PromotionalText VARCHAR2(100)
BookID INT NOT NULL,
CONSTRAINT PriceOffers_PK PRIMARY KEY (PriceOfferID),
CONSTRAINT PriceOffers_FK1 FOREIGN KEY (BookID) REFERENCES Books (BookID)
);


CREATE TABLE Authors (
AuthorID INT NOT NULL,
Name VARCHAR2(30),
CONSTRAINT Authors _PK PRIMARY KEY (AuthorID)
);


CREATE TABLE Review (
ReviewID INT NOT NULL,
VoterName VARCHAR2(40),
NumStars INT NOT NULL,
Comment VARCHAR2(300),
BookID INT NOT NULL,
CONSTRAINT Review_PK PRIMARY KEY (ReviewID),
CONSTRAINT Review_FK1 FOREIGN KEY (BookID) REFERENCES Books (BookID)
);
```

2). Create the table BookAuthor without definition of keys. [5]

ANSWER:

```
CREATE TABLE BookAuthor (
BookID INT NOT NULL,
AuthorID INT NOT NULL,
Order VARCHAR2 (30)
);
```

3). Add PK and FK to the table BookAuthor [5]

ANSWER:

```
ALTER TABLE BookAuthor (
ADD CONSTRAINT BookAuthor_PK PRIMARY KEY (BookID, AuthorID),
ADD CONSTRAINT BookAuthor_FK1 FOREIGN KEY (BookID) REFERENCES Books
(BookID),
```

```
ADD CONSTRAINT BookAuthor_FK2 FOREIGN KEY (AuthorID) REFERENCES Authors
(AuthorID)
);
```

4). Drop table "Authors" [5]

ANSWER:

```
ALTER TABLE BookAuthor DROP CONSTRAINT BookAuthor_FK2;
DROP TABLE Authors
```

5). Add a new attribute "ISBN" to the table "Books", and set it as primary key [5]

ANSWER:

As Books table already has a Primary key (BookID), we can solve this question in 2 ways.

1st WAY:

Removing BookID as Primary key and setting ISBN as a new primary key in the Books Table as follows:

(In order to drop primary key (BookID) in books table, first we need to drop bookID in other tables that has been set as foreign keys):

```
ALTER TABLE PriceOffers DROP CONSTRAINT PriceOffers_FK1;
ALTER TABLE Review DROP CONSTRAINT Review_FK1;
ALTER TABLE BookAuthor DROP CONSTRAINT BookAuthor_FK1;
```

(Then drop primary key (BookID) in Books table):

```
ALTER TABLE Books
DROP CONSTAINT Books_PK;
```

(Then add new attribute ISBN and set it as a new PK in books table):

```
ALTER TABLE Books
ADD COLUMN ISBN INT NOT NULL,
ADD CONSTRAINT Books_PK PRIMARY KEY (ISBN);
```

(To have a connection with other tables we need to set ISBN as a FK in other tables)

```
ALTER TABLE PriceOffers
ADD CONSTRAINT PriceOffers_FK1 FOREIGN KEY (ISBN) REFERENCES Books (ISBN);

ALTER TABLE Review
ADD CONSTRAINT Review_FK1 FOREIGN KEY (ISBN) REFERENCES Books (ISBN);
```

```
ALTER TABLE BookAuthor
ADD CONSTRAINT BookAuthor_FK1 FOREIGN KEY (ISBN) REFERENCES Books
(ISBN);
```

2nd WAY:

Keeping BookID as Primary key and setting BookID and ISBN together as a primary key in the Books Table as follows:

(To set BookID and ISBN together as a pk in books table, first we need to drop bookid alone as a pk in books table. To do so, we need to remove bookid in other tables that has been set as a fk's):

```
ALTER TABLE PriceOffers DROP CONSTRAINT PriceOffers_FK1;
ALTER TABLE Review DROP CONSTRAINT Review_FK1;
ALTER TABLE BookAuthor DROP CONSTRAINT BookAuthor_FK1;
```

(Then drop BookID as a Pk in books table):

```
ALTER TABLE Books
DROP CONSTAINT Books_PK;
```

(Then add new column ISBN and set both bookID and ISBN together as a PK in books table):

```
ALTER TABLE Books
ADD ISBN INT NOT NULL,
ADD CONSTRAINT Books_PK PRIMARY KEY (BookID,ISBN);
```

(As we are keeping BookID as one of the pk's in book table, we need to re-assign it as a FK in other tables that we dropped previously. To do so, we need to alter these tables again and re-add bookID as a FK in these tables):

```
ALTER TABLE PriceOffers
ADD CONSTRAINT PriceOffers_FK1 FOREIGN KEY (BookID) REFERENCES Books
(BookID);

ALTER TABLE Review
ADD CONSTRAINT Review_FK1 FOREIGN KEY (BookID) REFERENCES Books
(BookID);

ALTER TABLE BookAuthor
ADD CONSTRAINT BookAuthor_FK1 FOREIGN KEY (BookID) REFERENCES Books
(BookID);
```

6). In the table "Books", we have the attribute "publisher" which refers to the publisher name. Now, we want to have a new entity "Publisher" with attributes, such as PublisherID, PublisherName, Address, Country, Tel, ContactPersonName, ContactPersonalEmail. Use DDL to make relevant changes. [5]

ANSWER:

(create a new table called Publisher and add constraints in it):

```
CREATE TABLE Publisher (
PublisherID INT NOT NULL,
PublisherName VARCHAR2(50),
Address VARCHAR2(200),
Country VARCHAR2(30),
Tel INT,
ContactPersonName VARCHAR2(50),
ContactPersonalEmail VARCHAR2(40)
ADD CONSTRAINT Publisher_PK PRIMARY KEY (PublisherID)
ADD CONSTRAINT Publisher_FK1 FOREIGN KEY (BookID) REFERENCES)
Books(BookID)   (#one way of adding FK here – adding bookid as FK in publisher table)
);
```

(Other way of adding FK here): (#adding publisher id as FK in books table)

```
ALTER TABLE Books
ADD COLUMN PublisherID INT NOT NULL,
ADD CONSTRAINT Books_FK1 FOREIGN KEY (PublisherID) REFERENCES
Publisher(PublisherID);
```

7). Add default value 9.9 to "Price" in table "Books" [5]

ANSWER:

We can write the syntax in different ways according to the professor ppt slides:

Way 1:

```
ALTER TABLE Books
MODILFY Price FLOAT DEFAULT '9.9';
```

Way 2:

```
ALTER TABLE Books
ALTER COLUMN Price
SET DEFAULT '9.9';
```

Way 3:

```
ALTER TABLE Books
SET Price DEFAULT '9.9';
```

8). Update the datatype as varchar(200) for ImageUrl in table "Books" [5]

ANSWER:

We can solve this in 2 ways:

1st WAY:

```
ALTER TABLE Books
MODIFY COLUMN ImageUrl VARCHAR2(200);
```

2nd WAY:

```
ALTER TABLE Books
ADD COLUMN ImageUrl2 VARCHAR2(200);          #adding a temporary column ImageUrl2

UPDATE Books
SET ImageUrl2 = TO_CHAR(ImageUrl);                    #Copy values from ImageUrl

ALTER TABLE Books
DROP COLUMN ImageUrl;                                 #drop old column ImageUrl

ALTER TABLE Books
RENAME COLUMN ImageUrl2 to ImageUrl;                  #Rename column to ImageUrl
```

2. By using the sample of the data tables below, using SQL to answer the following questions [40]

Note: the tables below just gave you sample data, you should assume that there are many more rows in each table. These sample rows are just used for you to better understand the values in each table. You do not need to run them in Oracle

TUTOR (TutorID, CertDate, Status)

TutorID	CertDate	Status
100	1/05/2018	Active
101	1/05/2018	Temp Stop
102	1/05/2018	Dropped
103	5/22/2018	Active
104	5/22/2018	Active
105	5/22/2018	Temp Stop
106	5/22/2018	Active

STUDENT (StudentID, Group, Read)

StudentID	Group	Read
3000	3	2.3
3001	2	5.6
3002	3	1.3
3003	1	3.3
3004	2	2.7
3005	4	4.8
3006	3	7.8
3007	4	1.5

MATCH HISTORY (MatchID, TutorID, StudentID, StartDate, EndDate)

MatchID	TutorID	StudentID	StartDate	EndDate
1	100	3000	1/10/2018	
2	101	3001	1/15/2018	5/15/2018
3	102	3002	2/10/2018	3/01/2018
4	106	3003	5/28/2018	
5	103	3004	6/01/2018	6/15/2018
6	104	3005	6/01/2018	6/28/2018
7	104	3006	6/01/2018	

1) how many tutors have a status of temp stop? Which tutors are active?

ANSWER:

```
SELECT COUNT (TutorID)          (As it is asked 'how many tutors', we need to use 'count')
FROM Tutor
WHERE Status = 'Temp stop';
```

```
SELECT TutorID
FROM Tutor
WHERE Status = 'Active';
```

2) List the IDs of the tutors who are currently tutoring more than 1 student.

ANSWER:

```
SELECT TutorID
FROM Match History
GROUP BY TutorID
HAVING COUNT(StudentID) > 1;
```

(having is a conditional expression used to assign conditions on groups)

3) Get the list of student groups and also return the number of students in each group.

ANSWER:

```
SELECT Group, COUNT (*)
FROM Student
GROUP BY Group;

Select group, count(studentid)
From student
Group by group
```

(Or)

```
SELECT StudentID, COUNT (*)
FROM Student
GROUP BY StudentID
HAVING COUNT (Group) >1;
```

4) Which student has the highest Read score? And what is the score for that.

ANSWER:

We can solve this in 2 ways by using order by statement:

1st way :

```
SELECT StudentID, Read
FROM Student
ORDER BY Read DESC
FETCH FIRST 1 ROW ONLY;
```

2nd way:

```
SELECT StudentID, MAX(Read)
FROM Student
GROUP BY StudentID
```

```
ORDER BY MAX(Read) DESC
FETCH FIRST 1 ROW ONLY;
```

5) Show the average, max, min Read scores per student group.

ANSWER:

```
SELECT Group, AVG(Read) AS “AVERAGE”, MAX(Read), MIN(Read)
FROM Student
GROUP BY Group;
```

3. use the EnrollSys database (W6_Enrollment.sql) we introduced in the class, and write down the SQL statements to answer the following questions. [20]

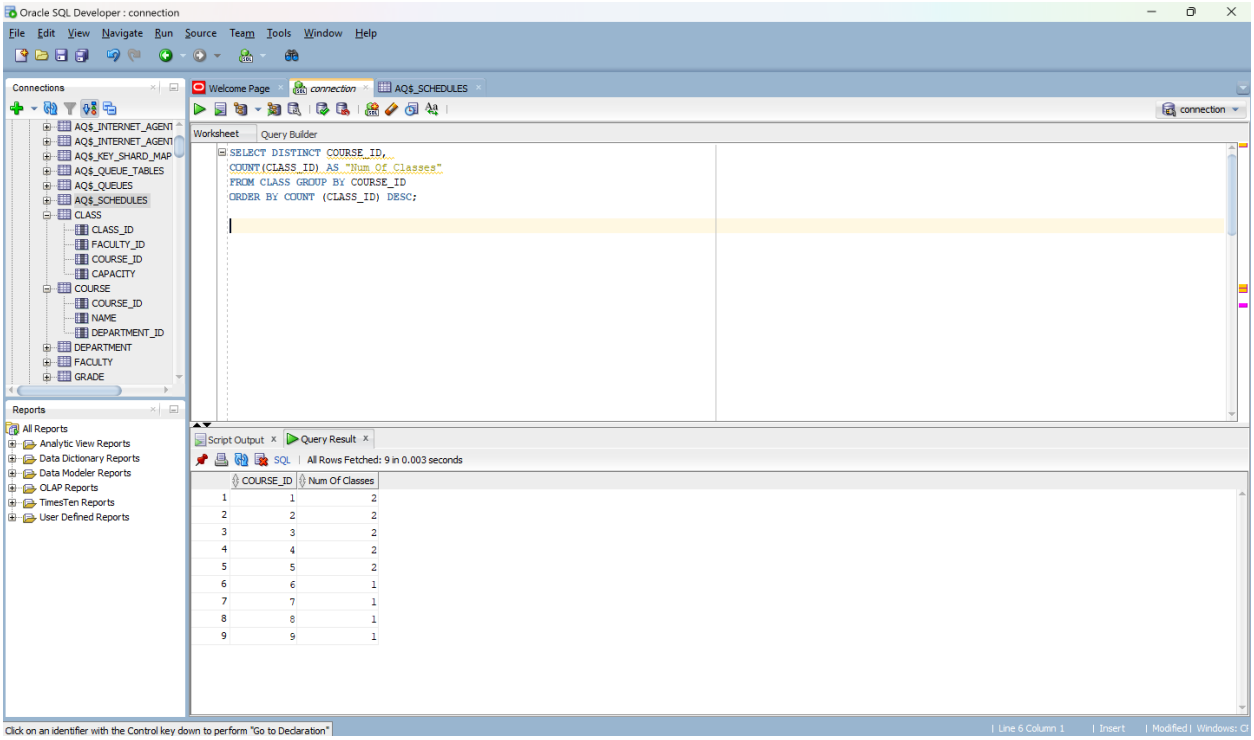
Note:

- You should provide your SQL statement, as well as the snapshot of the outputs in your Oracle
- Use a single SQL query for the following questions.
- Note that you should have the capability to write down correct SQL statement without running/examining them in the Oracle.

1). Return a list of unique courses along with the number of classes associated with it, rank the courses by the number of classes in descending order. Note: a class can be considered as a section in a course

ANSWER:

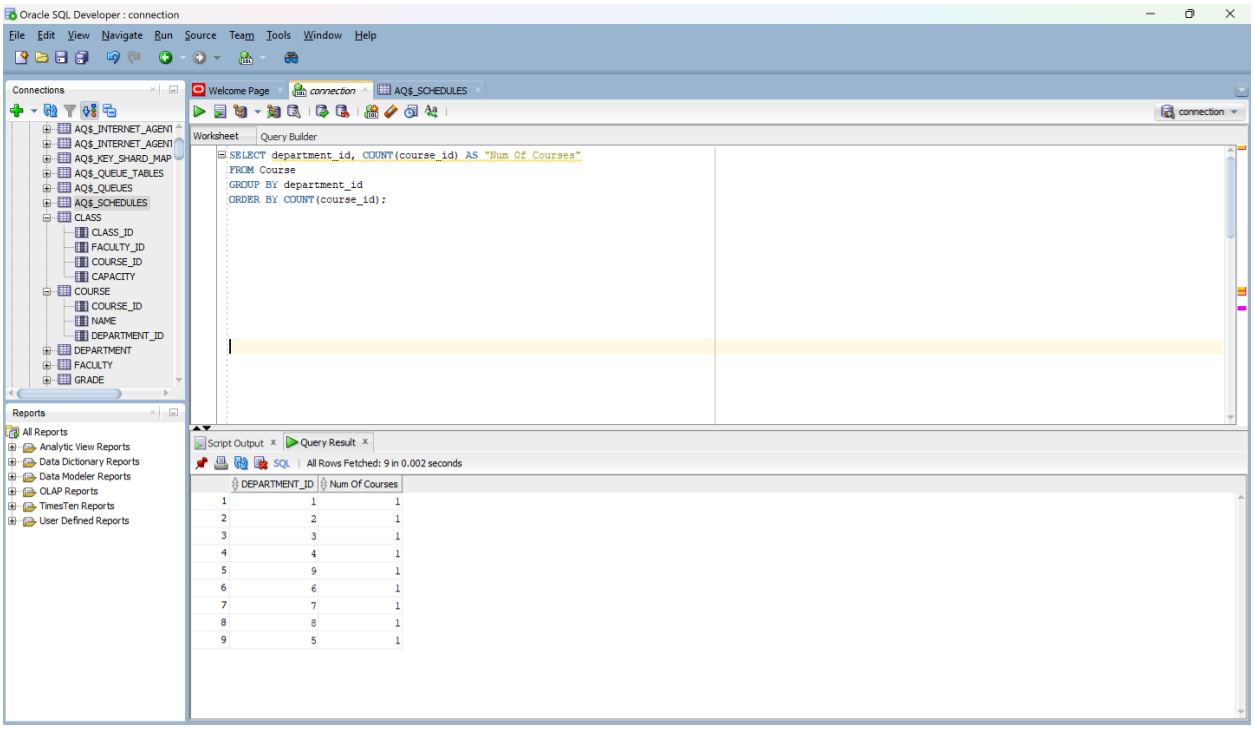
```
SELECT DISTINCT COURSE_ID, COUNT(CLASS_ID) AS "Num Of Classes"
FROM CLASS
GROUP BY COURSE_ID
ORDER BY COUNT (CLASS_ID) DESC;
```



2. Return a list of unique departments along with the number of courses operated by the department.

ANSWER:

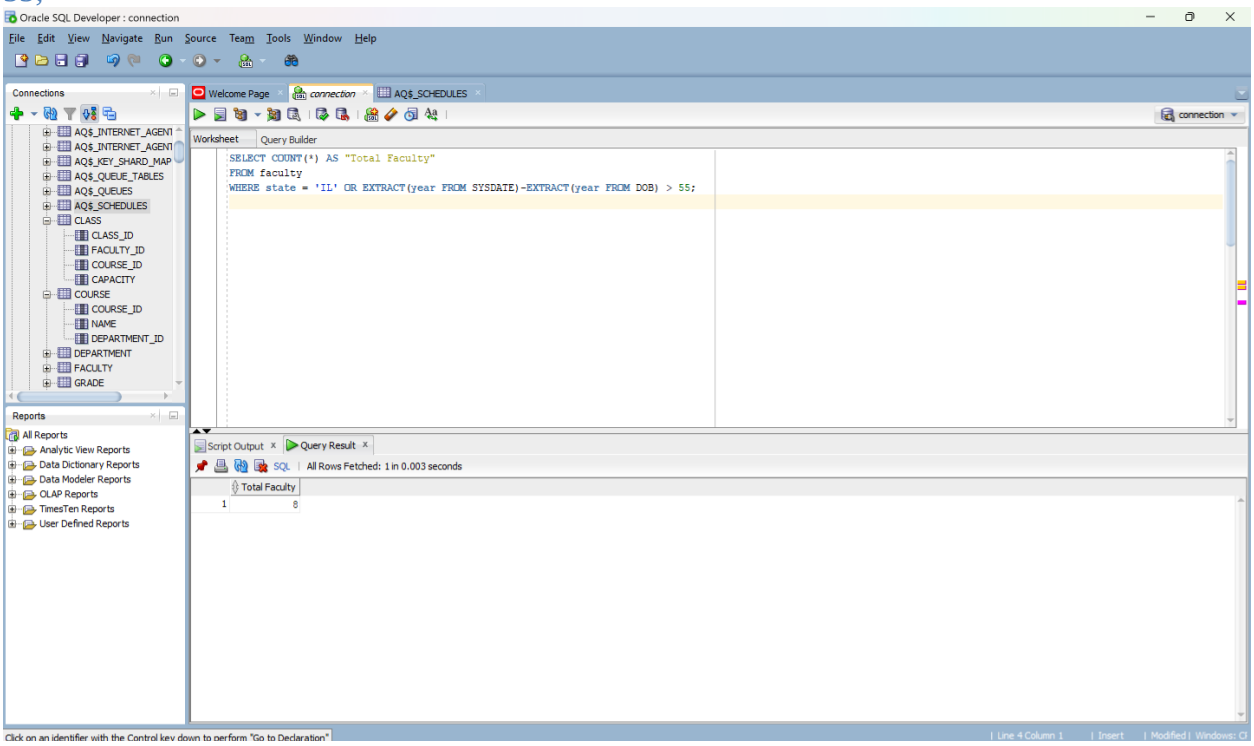
```
SELECT distinct department_id, COUNT(course_id) AS "Num Of Courses"
FROM Course
GROUP BY department_id
ORDER BY COUNT(course_id);
```



3. Find how many faculties are from IL, or age is > 55.

ANSWER:

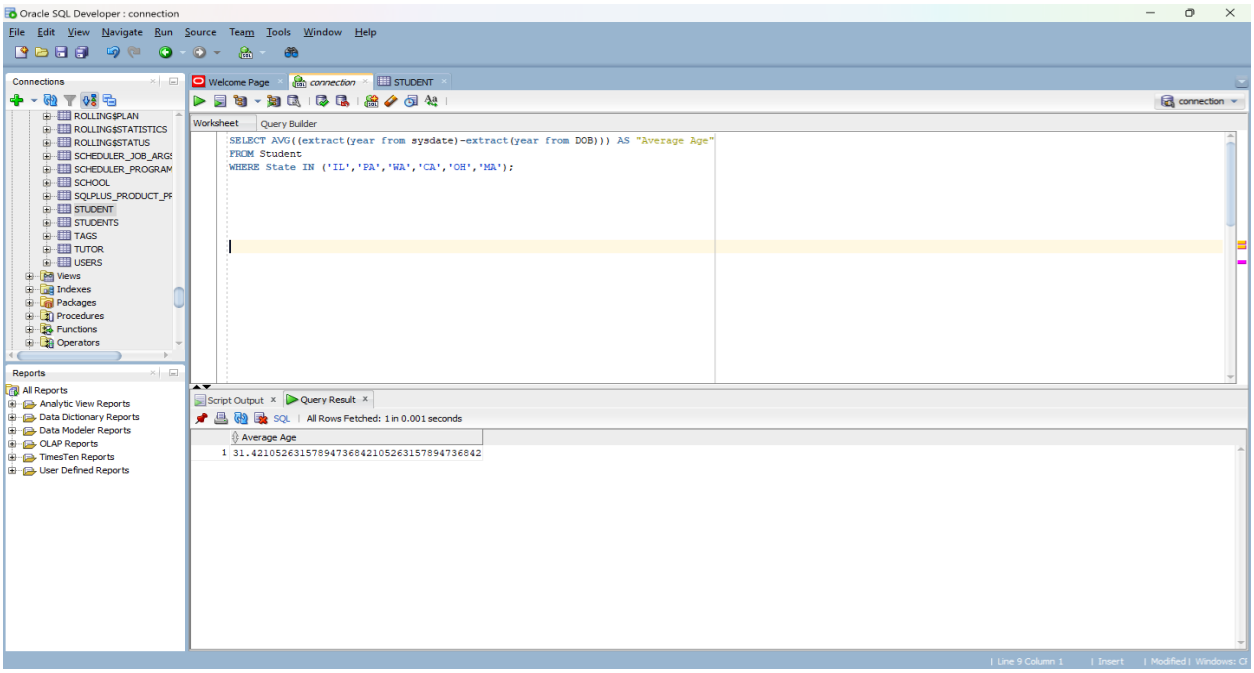
```
SELECT COUNT(*) AS "Total Faculty"
FROM faculty
WHERE state = 'IL' OR EXTRACT(year FROM SYSDATE)-EXTRACT(year FROM DOB) > 55;
```



4. Find the average age for students who are from IL, PA, WA, CA, OH, MA

ANSWER:

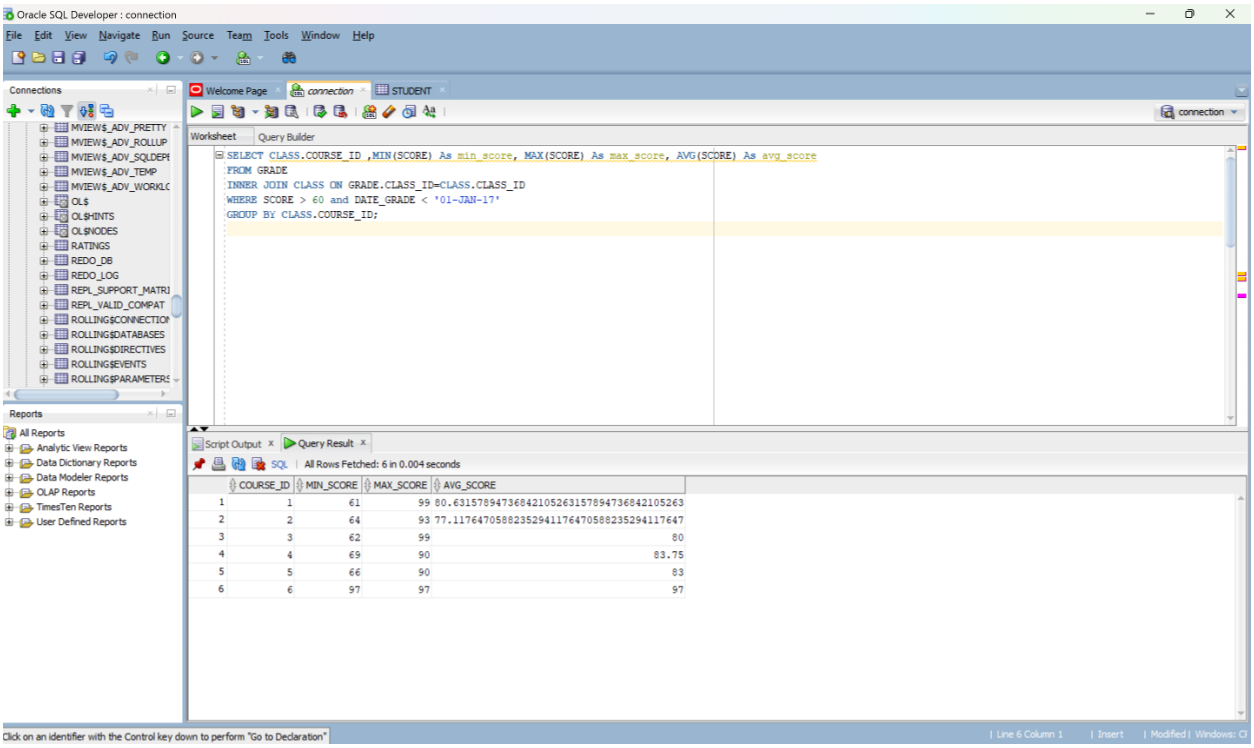
```
SELECT AVG((extract(year from sysdate)-extract(year from DOB))) AS "Average Age"
FROM Student
WHERE State IN ('IL','PA','WA','CA','OH','MA');
```



5. return the min, max and avg score of the unique courses, only for the courses graded before 2017, and the min grade is > 60.

ANSWER:

```
SELECT CLASS.COURSE_ID ,MIN(SCORE) As "min_score", MAX(SCORE) As
"max_score", AVG(SCORE) As "avg_score"
FROM GRADE
INNER JOIN CLASS ON GRADE.CLASS_ID=CLASS.CLASS_ID
WHERE GRADE.SCORE > 60 and (EXTRACT(YEAR FROM
GRADE.DATE_GRADE))<2017
GROUP BY CLASS.COURSE_ID;
```



Feedback & score: I got 97. (updated from 92 to 95) but in the price default question don't use quotation marks for value 9.9 as its data type is float.

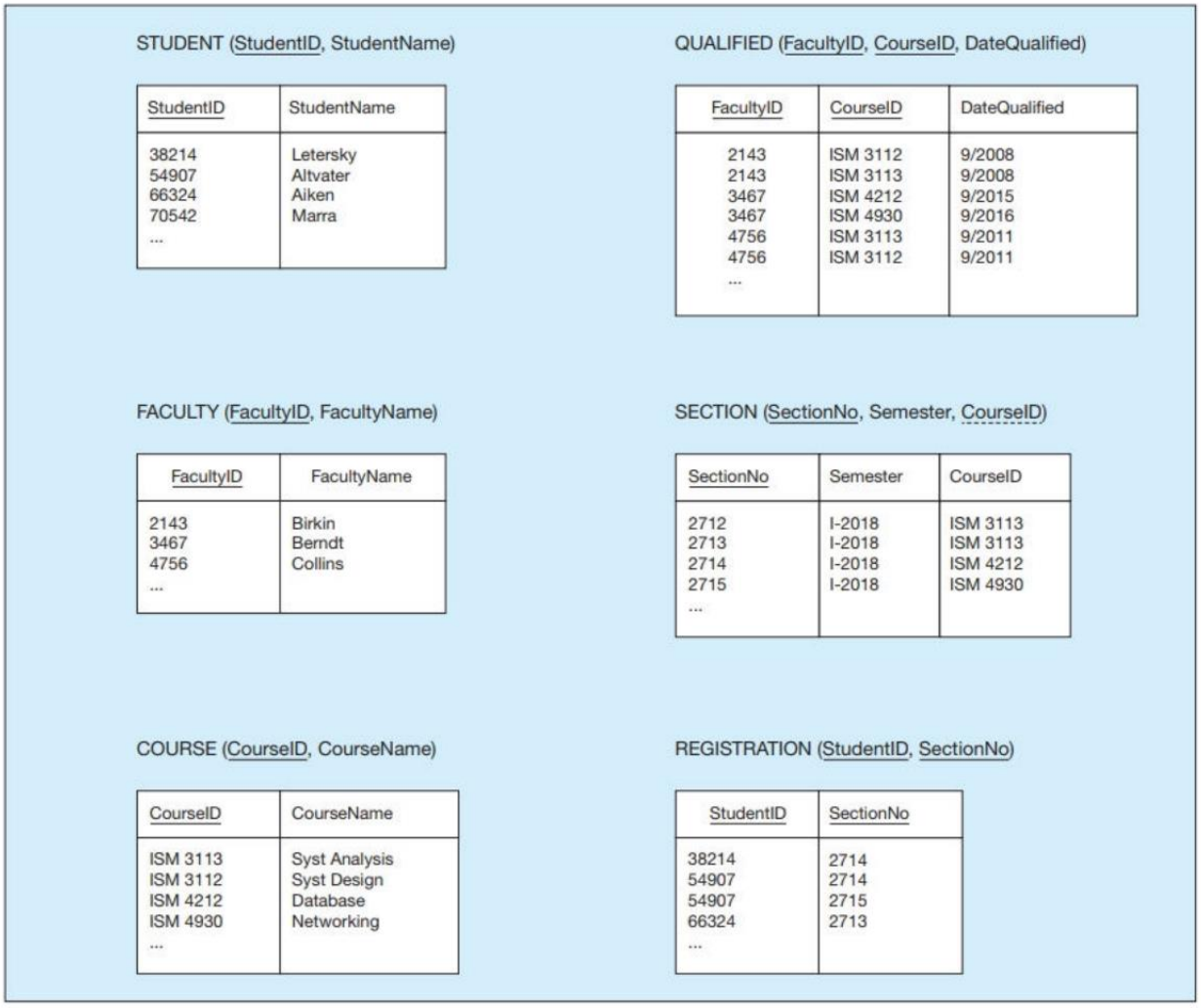
Q3. 5. -3 Incorrect query. Your query uses the WHERE clause to filter the results based on the score being greater than 60 and the date being before '01-JAN-2017'. However, it does not filter the results based on the minimum score being greater than 60 after grouping the results by course name.

Homework 6

Your Name: Naga Satya Silpa Annadevara

Student ID: A20517818

Q1. Use SQL to answer the following questions by referring to the DB shown below



STUDENT (StudentID, StudentName)

StudentID	StudentName
38214	Letersky
54907	Altvater
66324	Aiken
70542	Marra
...	

QUALIFIED (FacultyID, CourseID, DateQualified)

FacultyID	CourseID	DateQualified
2143	ISM 3112	9/2008
2143	ISM 3113	9/2008
3467	ISM 4212	9/2015
3467	ISM 4930	9/2016
4756	ISM 3113	9/2011
4756	ISM 3112	9/2011
...		

FACULTY (FacultyID, FacultyName)

FacultyID	FacultyName
2143	Birkin
3467	Berndt
4756	Collins
...	

SECTION (SectionNo, Semester, CourseID)

SectionNo	Semester	CourseID
2712	I-2018	ISM 3113
2713	I-2018	ISM 3113
2714	I-2018	ISM 4212
2715	I-2018	ISM 4930
...		

COURSE (CourseID, CourseName)

CourseID	CourseName
ISM 3113	Syst Analysis
ISM 3112	Syst Design
ISM 4212	Database
ISM 4930	Networking
...	

REGISTRATION (StudentID, SectionNo)

StudentID	SectionNo
38214	2714
54907	2714
54907	2715
66324	2713
...	

1. write SQL to answer the following questions [15]

a. Display the course ID and course name for all courses with an ISM prefix.

b. Display the numbers and names of all courses for which Professor Berndt has been qualified.

c. Display the class roster, including student name, for all students enrolled in section 2714 of ISM 4212.

ANSWER:

a.

```
SELECT CourseID, CourseName
FROM COURSE
WHERE CourseID LIKE  '%ISM';
```

b.

```
SELECT COURSE. CourseID, COURSE. CourseName
FROM COURSE, FACULTY, QUALIFIED
WHERE COURSE. CourseID = QUALIFIED. CourseID
AND FACULTY.FacultyID = QUALIFIED. FacultyID
AND FACULTY.FacultyName = 'Berndt';
```

(Or)
(We can use 'AS' (Alias) to shorten the query & improve readability):

```
SELECT c.CourseID, c.CourseName
FROM COURSE AS c, FACULTY AS f, QUALIFIED AS q
WHERE c. CourseID = q. CourseID
AND f.FacultyID = q. FacultyID
AND f.FacultyName = 'Berndt';
```

(Or)
(We can use join statement):

```
SELECT c.CourseID, c.CourseName
FROM Qualified q
INNER JOIN COURSE c ON c.CourseID = q.CourseId
INNER JOIN FACULTY f ON f.FacultyID = q.FacultyID
WHERE f.FacultyName  = 'Berndt';
```

c.
(Using 'AS' to shorten the query & improve readability):

```
SELECT s.StudentID, s.StudentName, sn.CourseID, r.SectionNo
FROM STUDENT AS s, SECTION AS sn, REGISTRATION AS r
WHERE s.StudentID = r.StudentID
AND sn.SectionNo = r.SectionNo
AND sn.CourseID = 'ISM 4214'
AND r.Section No = '2714';
```

(Or)
(We can use join statement):

```
SELECT s.StudentID, s.StudentName, sn.CourseID, r.SectionNo
FROM REGISTRATION r
INNER JOIN STUDENT s ON s.StudentID = r.StudentID
INNER JOIN SECTION sn ON sn.SectionNo = r.SectionNo
WHERE r.SectionNo = 2714;
```

2. write SQL statements [25]

a. What are the names of the course(s) that student Altvater took during the semester I-2018?
b. List the names of the students who have taken at least one course that Professor Collins is qualified to teach.
c. List the names of the students who took at least one course with "Syst" in its name during the semester I-2018.
d. How many students did Professor Collins teach during the semester I-2018?
e. List the names of the courses that at least two faculty members are qualified to teach.

ANSWER:

a.

```
SELECT COURSE.CourseName
FROM COURSE, STUDENT, SECTION, REGISTRATION
WHERE STUDENT.StudentID = REGISTRATION. StudentID
AND COURSE.CourseID = SECTION.CourseID
AND SECTION.SectionNo = REGISTRATION. SectionNo
AND STUDENT.StudentName = 'Altvater'
AND SECTION.Semester = 'I-2018';
```

(Or)
(We can use 'AS' (Alias) to shorten the query & improve readability):

```
SELECT c.CourseName
FROM COURSE AS c, STUDENT AS s, SECTION AS sn, REGISTRATION AS r
WHERE s.StudentID = r. StudentID
AND c.CourseID =sn.CourseID
AND sn.SectionNo = r. SectionNo
AND s.StudentName = 'Altvater'
AND sn.Semester = 'I-2018';
```

(Or)
(We can use join statement):

```
SELECT c.CourseName
FROM STUDENT s
NATURAL JOIN REGISTRATION r
NATURAL JOIN SECTION sn
NATURAL JOIN COURSE c
WHERE s.StudentName = 'Altvater' AND sn.Semester = 'I-2018';
```

(Note: We can use natural join here as it automatically compares the attributes with same names. So here, as we have same column names, we can use natural join to reduce the confusion)

b. (as it is asked 'at least one course', we need to use distinct as 'distinct' eliminates duplicate values)

```
SELECT  DISTINCT  STUDENT.StudentName
FROM  STUDENT, FACULTY, REGISTRATION, SECTION, QUALIFIED
WHERE STUDENT.StudentID =  REGISTRATION. StudentID
AND  FACULTY.FacultyID = QUALIFIED. FacultyID
AND  REGISTRATION.SectionNo = SECTION. SectionNo
AND  SECTION.CourseID =  QUALIFIED. CourseID
AND  FACULTY.FacultyName = 'Collins';
```

(Or)
(We can use 'AS' (Alias) to shorten the query & improve readability):

```
SELECT  DISTINCT  s.StudentName
FROM  STUDENT AS s, FACULTY AS f, REGISTRATION AS r, SECTION AS
sn, QUALIFIED AS q
WHERE  s.StudentID =  r. StudentID
AND  f.FacultyID = q. FacultyID
AND  r.SectionNo = sn. SectionNo
AND  sn.CourseID =  q. CourseID
AND  f.FacultyName = 'Collins';
```

(Or)
(We can use join statement):

```
SELECT DISTINCT s.StudentName
FROM STUDENT s
NATURAL JOIN REGISTRATION r
NATURAL JOIN SECTION sn
NATURAL JOIN COURSE c
NATURAL JOIN FACULTY f
NATURAL JOIN QUALIFIED q
WHERE f.FacultyName = 'Collins';
```

(Note: We can use natural join here as the it automatically compares the attributes with same names. So here, as we have same column names, we can use natural join to reduce the confusion)

c. (as it is asked 'at least one course', we need to use distinct as distinct eliminates duplicate values)

```
SELECT  DISTINCT  STUDENT.StudentName
FROM STUDENT, COURSE, SECTION, REGISTRATION
WHERE  STUDENT.StudentID = REGISTRATION. StudentID
AND  COURSE.CourseID = SECTION. CourseID
AND  REGISTRATION.SectionNo = SECTION. SectionNo
AND   COURSE.CourseName LIKE   '%Syst%'              (# syst can be at start/end)
AND   SECTION.Semester = 'I-2018';
```

(Or)
(We can use 'AS' (Alias) to shorten the query & improve readability):

```
SELECT  DISTINCT  s.StudentName
FROM STUDENT AS s, COURSE AS c, SECTION AS sn, REGISTRATION AS r
WHERE  s.StudentID = r. StudentID
AND  c.CourseID = sn. CourseID
AND  r.SectionNo = sn. SectionNo
AND   c.CourseName LIKE   '%Syst%'              (# syst can be at start/end)
AND   sn.Semester = 'I-2018';
```

(Or)
(We can use join statement):

```
SELECT DISTINCT s.StudentName
FROM STUDENT s
NATURAL JOIN REGISTRATION r
NATURAL JOIN SECTION sn
NATURAL JOIN COURSE c
WHERE c.CourseName LIKE '%Syst%' AND sn.Semester='I-2018';
```

(Note: We can use natural join here as the it automatically compares the attributes with same names. So here, as we have same column names, we can use natural join to reduce the confusion)

d. (As it is asked 'how many students', we need to use count(student_id)):

```
SELECT  COUNT (REGISTRATION_StudentID) AS  Num_Of_Students
FROM     REGISTRATION, SECTION, QUALIFIED, FACULTY
WHERE  FACULTY.FacultyID =  QUALIFIED. FacultyID
AND    REGISTRATION.SectionNo =  SECTION. SectionNo
AND    QUALIFIED.CourseID =  SECTION. CourseID
```

```
AND    FACULTY.FacultyName = 'Collins'
AND    SECTION.Semester = 'I-2018';
```

(Or)
(We can use 'AS' (Alias) to shorten the query & improve readability):

```
SELECT  COUNT (r.StudentID) AS  Num_Of_Students
FROM    REGISTRATION AS r, SECTION AS sn, QUALIFIED AS q, FACULTY
AS f
WHERE  f.FacultyID =  q. FacultyID
AND    r.SectionNo =  sn. SectionNo
AND    q.CourseID =  sn. CourseID
AND    f.FacultyName = 'Collins'
AND    sn.Semester = 'I-2018';
```

(Or)
(We can use join statement):

```
SELECT COUNT(r.StudentID)
FROM REGISTRATION r
NATURAL JOIN SECTION sn
NATURAL JOIN QUALIFIED q
NATURAL JOIN FACULTY f
WHERE f.FacultyName = 'Collins' AND sn.Semester = 'I-2018';
```

(Note: We can use natural join here as the it automatically compares the attributes with same names. So here, as we have same column names, we can use natural join to reduce the confusion)

(Or)
(We can use sub query by using in statement):

```
SELECT Count(StudentID)
FROM REGISTRATION
WHERE SectionNo
IN (SELECT SectionNo FROM SECTION WHERE CourseID IN (SELECT
CourseID FROM QUALIFIED WHERE FacultyID = (SELECT FacultyID FROM
FACULTY WHERE FacultyName = 'Collins'))
AND Semester = 'I-2018');
```

e.

```
SELECT COURSE.CourseName
FROM COURSE, QUALIFIED
WHERE COURSE.CourseID = QUALIFIED.CourseID
GROUP BY CourseID
```

HAVING COUNT (FacultyID) >=2;

(Or)
(We can use 'AS' (Alias) to shorten the query & improve readability):

```
SELECT c.CourseName
FROM COURSE AS c, QUALIFIED AS  q
WHERE c.CourseID = q.CourseID
GROUP BY CourseID
HAVING COUNT (FacultyID) >=2;
```

Q2. Provide the outputs by the following SQL statements [30]

TUTOR (TutorID, CertDate, Status)

TutorID	CertDate	Status
100	1/05/2018	Active
101	1/05/2018	Temp Stop
102	1/05/2018	Dropped
103	5/22/2018	Active
104	5/22/2018	Active
105	5/22/2018	Temp Stop
106	5/22/2018	Active

STUDENT (StudentID, Group, Read)

StudentID	Group	Read
3000	3	2.3
3001	2	5.6
3002	3	1.3
3003	1	3.3
3004	2	2.7
3005	4	4.8
3006	3	7.8
3007	4	1.5

MATCH HISTORY (MatchID, TutorID, StudentID, StartDate, EndDate)

MatchID	TutorID	StudentID	StartDate	EndDate
1	100	3000	1/10/2018	
2	101	3001	1/15/2018	5/15/2018
3	102	3002	2/10/2018	3/01/2018
4	106	3003	5/28/2018	
5	103	3004	6/01/2018	6/15/2018
6	104	3005	6/01/2018	6/28/2018
7	104	3006	6/01/2018	

TUTOR REPORT (MatchID, Month, Hours, Lessons)

MatchID	Month	Hours	Lessons
1	6/18	8	4
4	6/18	8	6
5	6/18	4	4
4	7/18	10	5
1	7/18	4	2

1. SELECT Tutor.TutorID, status, StudentID
   FROM Tutor Left Join MatchHistory
   ON Tutor.TutorID = MatchHistory.TutorID
   Where Tutor.TutorID = 104;

ANSWER: The output of the above query will be:

TutorID	Status	StudentID
104	Active	3005
104	Active	3006

2. SELECT Student.StudentID, Read
From Student Left Join MatchHistory
ON Student.StudentID = MatchHistory.StudentID
Where TutorID is NULL

ANSWER: The output of the above query will be:

StudentID	Read
3007	1.5

3. SELECT Student.StudentID, Read, Tutor.TutorID, Status
From student
Left Join MatchHistory ON Student.StudentID = MatchHistory.StudentID
Left Join Tutor ON Tutor.TutorID = MatchHistory.TutorID
Where Student.StudentID > 3004;

ANSWER: The output of the above query will be:

StudentID	Read	TutorID	Status
3005	4.8	104	Active
3006	7.8	104	Active
3007	1.5	NULL	NULL

Q3. Write down SQL statements based on DB in Q2 [30]

0. List all active students in June by name. (Make up names and other data if you are actually building a prototype database.) Include the number of hours students received tutoring and how many lessons they completed.

1. For each student group, list the number of tutors who have been matched with that group.

2. List the total number of lessons taught in 2018 by tutors in each of the three Status categories (Active, Temp Stop, and Dropped).

3. Which tutors, by name, are available to tutor? Write the SQL query.

5. Write a SQL query to identify all students who have been matched in 2018 with a tutor whose status is Temp Stop.

6. Write the SQL query to find any tutors who have not submitted a report for July.

ANSWER:

0. As there is no ‘student name’ column in the student table, assume that we added ‘StudentName’ column in the student table. Then the query will be :

```
SELECT STUDENT.StudentID, STUDENT.StudentName, COUNT(TUTOR REPORT.Hours) AS Num_Of_hours, COUNT(TUTOR REPORT.Lessons) AS Num-Of_Lessons
FROM STUDENT, MATCH HISTORY, TUTOR REPORT
WHERE STUDENT.StudentID = MATCH HISTORY.StudentID
AND MATCH HISTORY.MatchID = TUTOR REPORT.MatchID
AND EXTRACT (MONTH FROM (MATCH HISTORY. EndDate) >= 6
```

(Or)
(We can use ‘AS’ (Alias) to shorten the query & improve readability):

```
SELECT s.StudentID, s.StudentName, COUNT(tr.Hours) AS Num_Of_hours, COUNT(tr.Lessons) AS Num-Of_Lessons
FROM STUDENT AS s, MATCH HISTORY AS mh, TUTOR REPORT AS tr
WHERE s.StudentID = mh.StudentID
AND mh.MatchID = tr.MatchID
AND EXTRACT (MONTH FROM (mh. EndDate) >= 6
```

(Or)
(We can extract month in other way by changing the last line of the query):

```
AND tr.Month LIKE '6%';
```

(Or)
(We can use join statement):

```
SELECT s.StudentID, s.StudentName, COUNT(tr.Hours) AS Num_Of_hours, COUNT(tr.Lessons) AS Num-Of_Lessons
FROM STUDENT AS s
INNER JOIN MATCH HISTORY AS mh ON s.StudentID = mh.StudentID
LEFT JOIN TUTOR REPORT AS tr ON mh.MatchID = tr.MatchID
WHERE mh.EndDate IS NULL
OR EXTRACT (MONTH FROM (mh. EndDate) >= 6
GROUP BY s.StudentID, s. StudentName;
```

1.

```
SELECT  STUDENT. Group, COUNT(MATCH HISTORY.TutorID) AS
Num_Of_Tutors
FROM STUDENT , MATCH HISTORY
WHERE  STUDENT. StudentID   =   MATCH HISTORY. StudentID
GROUP BY STUDENT.Group;
```

(Or)
(We can use 'AS' (Alias) to shorten the query & improve readability):

```
SELECT  s.StudentID , COUNT(mh.TutorID) AS Num_Of_Tutors
FROM STUDENT AS s , MATCH HISTORY AS mh
WHERE  s. StudentID   =   mh. StudentID
GROUP BY s.Group;
```

2.

```
SELECT COUNT (TUTOR REPORT. Lessons) AS Num_Of_Lessons,
TUTOR.TutorID, TUTOR.Status
FROM  TUTOR REPORT, TUTOR, MATCH HISTORY
WHERE TUTOR.TutorID = MATCH HISTORY. TutorID
AND TUTOR REPORT. MatchID =  MATCH HISTORY.MatchID
AND EXTRACT (YEAR FROM  (MATCH HISTORY. StartDate)) = 2018
OR EXTRACT (YEAR FROM (MATCH HISTORY.EndDate)) = 2018
GROUP BY TUTOR.Status
```

(Or)
(We can use 'AS' (Alias) to shorten the query & improve readability):

```
SELECT COUNT (tr. Lessons) AS Num_Of_Lessons  , t.TutorID,  t.Status
FROM  TUTOR REPORT AS tr, TUTOR AS t, MATCH HISTORY AS mh
WHERE t.TutorID = mh. TutorID
AND tr. MatchID =  mh.MatchID
AND EXTRACT (YEAR FROM  (mh. StartDate)) = 2018
OR EXTRACT (YEAR FROM (mh.EndDate)) = 2018
GROUP BY t.Status
```

(Or)
(We can use join statement):

```
SELECT t.Status, SUM(tr.Lessons) AS TotalLessons
FROM Tutor t
```

```
LEFT JOIN mh ON t.TutorID = mh.TutorID
LEFT JOIN tr ON mh.MatchID = tr.MatchID
WHERE EXTRACT(YEAR FROM TO_DATE(mh.StartDate, 'MM/DD/RRRR')) =
2018
GROUP BY t.Status;
```

3.

As there is no TutorName column in the Tutor table, assume that we added a new column called ‘TutorName’ to the Tutor table. Then the query will be:

```
SELECT  TutorID, Name, Status
FROM TUTOR
WHERE Status = ‘Active’;
```

5.

```
SELECT  MATCH HISTORY.StudentID
FROM  MATCH HISTORY, TUTOR
WHERE  MATCH HISTORY.TutodID  =  TUTOR.TutorID
AND  TUTOR .Status = ‘Temp Stop’
AND  EXTRACT (YEAR FROM (TUTOR.CertDate)) = 2018;
```

(Or)
(We can use ‘AS’ (Alias) to shorten the query & improve readability):

```
SELECT  mh.StudentID
FROM  MATCH HISTORY AS mh , TUTOR AS t
WHERE  mh.TutodID  =  t.TutorID
AND  t.Status = ‘Temp Stop’
AND  EXTRACT (YEAR FROM (t.CertDate)) = 2018;
```

6.

```
SELECT  MATCH HISTORY.TutorID
FROM  MATCH HISTORY, TUTOR REPORT
WHERE   MATCH HISTORY.MatchID =   TUTOR REPORT. MatchID
AND   TUTOR REPORT. Month  !=  ‘7/18’ ;
```

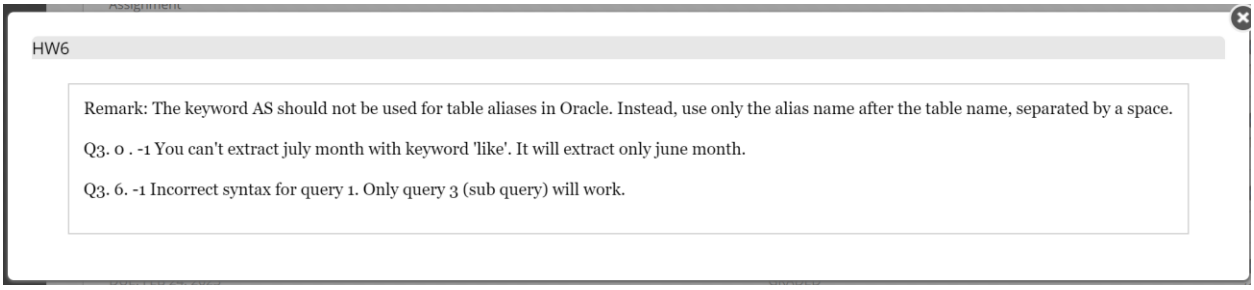
(Or)
(We can use ‘AS’ (Alias) to shorten the query & improve readability):

```
SELECT  mh.TutorID
FROM  MATCH HISTORY AS mh , TUTOR REPORT AS tr
WHERE   mh.MatchID =   tr. MatchID
AND   tr. Month  !=  ‘7/18’ ;
```

(Or)
(using join and subquery)

```
SELECT TutorID, Name
FROM Tutor
WHERE Status = 'Active' AND TutorID NOT IN
(SELECT TutorID FROM MatchHistory mh
INNER JOIN TutorReport tr ON tr.MatchID = mh.MatchID
WHERE EXTRACT(MONTH FROM TO_DATE(tr.Month, 'MM/DD')) = 7);
```

FEEDBACK & SCORE : 98 but for all the 3 questions, don't use 'as' for renaming table names as it don't work in oracle 21c. so the 2nd way is wrong in this assignment. 1st and join approach is correct.



Homework 7

Your Name: Naga Satya Silpa Annadevara

Student ID: A20517818

Q1. Assume we have a database which stores product and equipment information, as shown below. Use subqueries to answer the following questions [30]

```
Product(maker, model, type)
PC(model, speed, ram, hd, price)
Laptop(model, speed, ram, hd, screen, price)
Printer(model, color, type, price)
```

a) Find the makers of PC's with a speed of at least 3.0.

b) Find the printers with the highest price.

c) Find the laptops whose speed is slower than that of any PC.

d) Find the model number of the item (PC, laptop, or printer) with the highest price.

e) Find the maker of the color printer with the lowest price.

f) Find the maker(s) of the PC(s) with the fastest processor among all those PC's that have the smallest amount of RAM.

SOLUTION:

a)

```
SELECT DISTINCT maker
FROM Product
WHERE model IN (SELECT model FROM PC WHERE speed >= 3);
```

b)

```
SELECT model, price
FROM Printer
WHERE price IN (SELECT MAX(price) FROM Printer);
```

c)

```
SELECT model
FROM Laptop
WHERE Laptop.speed < (SELECT MIN(speed) FROM PC);
```

d)

```
WITH Model_Price AS (SELECT model, price
FROM PC
UNION
SELECT model, price
FROM Laptop
UNION
SELECT model, price
FROM Printer)
SELECT model
FROM Model_Price
WHERE price = (SELECT MAX(price) FROM Model_Price);
```

e)

```
SELECT  DISTINCT p.maker
FROM product p , printer r
WHERE p.model=r.model
AND r.price=(SELECT min(price) FROM printer);
```

Note: If the color column in the table specifies the type of print (i.e. color = true for a color printer and color = false for a black and white printer, we can include color=true in the where clause to select minimum price of only color printers.

f)

```
SELECT DISTINCT maker
FROM Product
WHERE model IN (SELECT model FROM PC WHERE speed=(SELECT
MAX(speed) FROM PC WHERE ram= (SELECT MIN(ram) FROM PC)));
```

Q2. Use appropriate SQL (regular, subqueries, joins, etc.) to answer the following questions by using the same database in Q1. Note, the “manufacturer” is referred as the “maker” in the table “product”. [30]

a) Find the average speed of PC's.

b) Find the average speed of laptops costing over $1000.

c) Find the average price of PC's made by manufacturer "A."

d) Find the average price of PC's and laptops made by manufacturer "D."

e) Find, for each different speed, the average price of a PC.

f) Find for each manufacturer, the average screen size of its laptops.

g) Find the manufacturers that make at least three different models of PC.

h) Find for each manufacturer who sells PC's the maximum price of a PC.

i) Find, for each speed of PC above 2.0, the average price.

j) Find the average hard disk size of a PC for all those manufacturers that make printers.

SOLUTION:

a)
```
SELECT AVG(speed) FROM PC;
```

b)
```
SELECT AVG(speed) FROM Laptop WHERE price>1000;
```

c)
```
SELECT AVG(price) AS "Avg_Price"
FROM Product p, PC c
WHERE p.model = c.model
AND p.maker='A';
```

d)
```
SELECT AVG(price) AS "Avg_Price"
FROM  (SELECT PC.price  FROM PC
INNER JOIN Product
ON PC.model = Product.model
```

```
WHERE  maker = 'D'
UNION ALL
SELECT Laptop.price
FROM   Laptop
INNER JOIN product
ON Laptop.model = Product.model
WHERE  maker = 'D');
```

e)

```
SELECT speed, AVG(price) AS “Avg_Price”
FROM PC
GROUP BY speed;
```

f)

```
SELECT p.maker, AVG(l.screen) AS “Avgscreen_size”
FROM product p
INNER JOIN laptop l
ON p.model=l.model
GROUP BY p.maker;
```

g)

```
SELECT maker, COUNT(DISTINCT model) AS “Number of models”
FROM product
WHERE type = 'PC'
GROUP BY maker
HAVING COUNT(DISTINCT model) >= 3;
```

h)

```
SELECT Product.maker,MAX(PC.price) AS “Max Price”
FROM Product
INNER JOIN PC
ON Product.model = PC.model
GROUP BY Product.maker ;
```

i)

```
SELECT speed, AVG(price) AS “Avg_Price”
FROM PC
WHERE speed >2.0
GROUP BY speed;
```

j)

```
SELECT p.maker, AVG(c.hd) AS “Average hardDisk Size”
```

```
FROM product p
JOIN PC c
ON p.model = c.model
WHERE p.maker IN (SELECT p.maker FROM product WHERE type='printer')
GROUP BY p.maker;
```

Q3. Using SQL to answer the following questions, by using the database in Q1 [20]

a) Using two `INSERT` statements, store in the database the fact that PC model 1100 is made by manufacturer C, has speed 3.2, RAM 1024, hard disk 180, and sells for $2499.

b) Insert the facts that for every PC there is a laptop with the same manufacturer, speed, RAM, and hard disk, a 17-inch screen, a model number 1100 greater, and a price $500 more.

c) Delete all PC's with less than 100 gigabytes of hard disk.

d) Delete all laptops made by a manufacturer that doesn't make printers.

e) Manufacturer A buys manufacturer B. Change all products made by B so they are now made by A.

SOLUTION:

a)

```
INSERT INTO Product VALUES ('C', 1100, 'PC')
INSERT INTO PC VALUES (1100 , 3.2, 1024, 180, 2499);
```

b)

```
INSERT INTO
   Product(maker, model, type) (
      SELECT
         maker,
         model + 1100,
         'Laptop'
      FROM
         Product
```

```
        WHERE
            type = 'PC'
    );


INSERT INTO
    Laptop(model, speed, ram, hd, screen, price) (
        SELECT
            model + 1000,
            speed,
            ram,
            hd,
            17,
            price + 500
        FROM
            PC
    );
```

c)

```
-- Delete records from both PRODUCT and PC Tables

DELETE *
FROM PRODUCT
WHERE MODEL IN (
          SELECT MODEL
          FROM PC WHERE hd<100
) AND TYPE = 'PC';

DELETE * FROM PC WHERE hd<100;
```

(Assuming that the hard disc can store the data in gigabytes)

d)

```
DELETE * FROM Laptop
WHERE model IN
(SELECT p.model  FROM product p,  laptop l
WHERE p.model = l.model
AND maker IN
((SELECT DISTINCT maker FROM product)
EXCEPT
(SELECT DISTINCT maker FROM product
WHERE type = 'printer')));
```

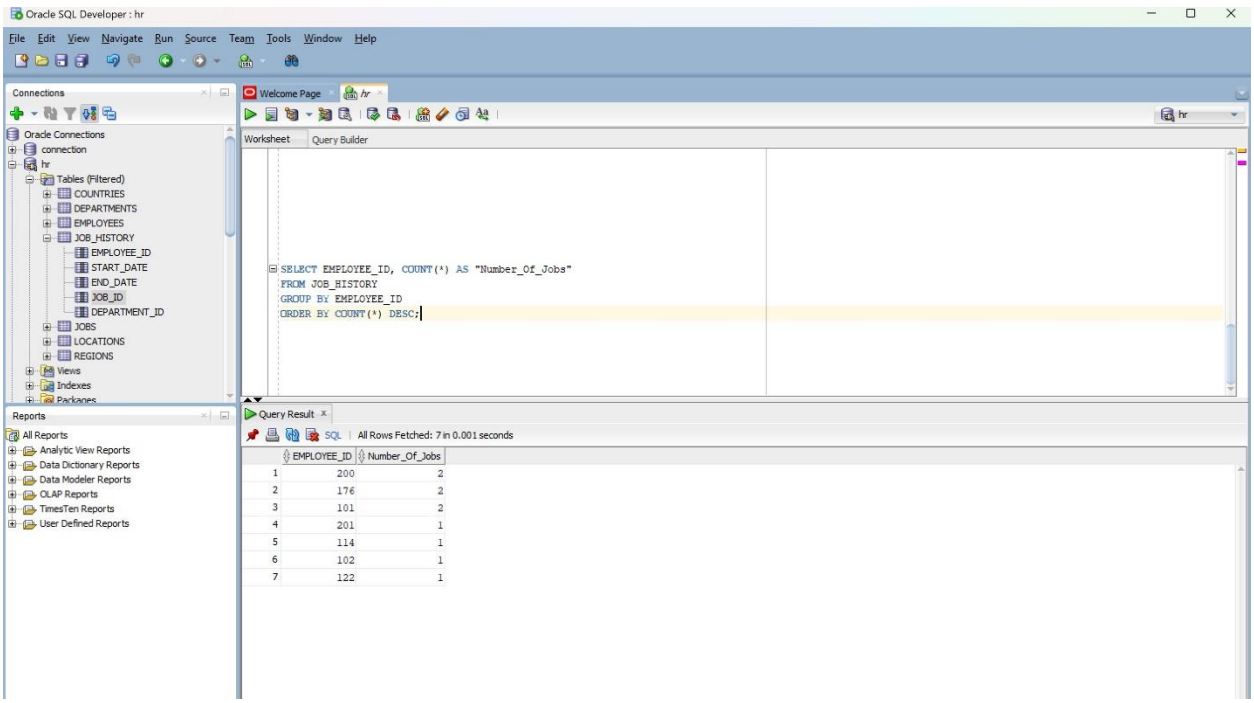
e)
```
UPDATE Product
SET maker =’A’
WHERE maker=’B’;
```

Q4. Give SQL for the following questions, and run them on the HR schema, give outputs in Oracle [20]

- Get employee ID and the number of his/her job histories, rank the outputs by the number of job histories in a descending way
- Return employee ID, name, emails, if he or she has more than 1 job history
- Return employee ID, name, and his or her all job titles
- Get department ID, name, and the number of employees in each department, sort records by the number of employees in descending order
- Get manager ID, name, and the number of employees he or she supervised

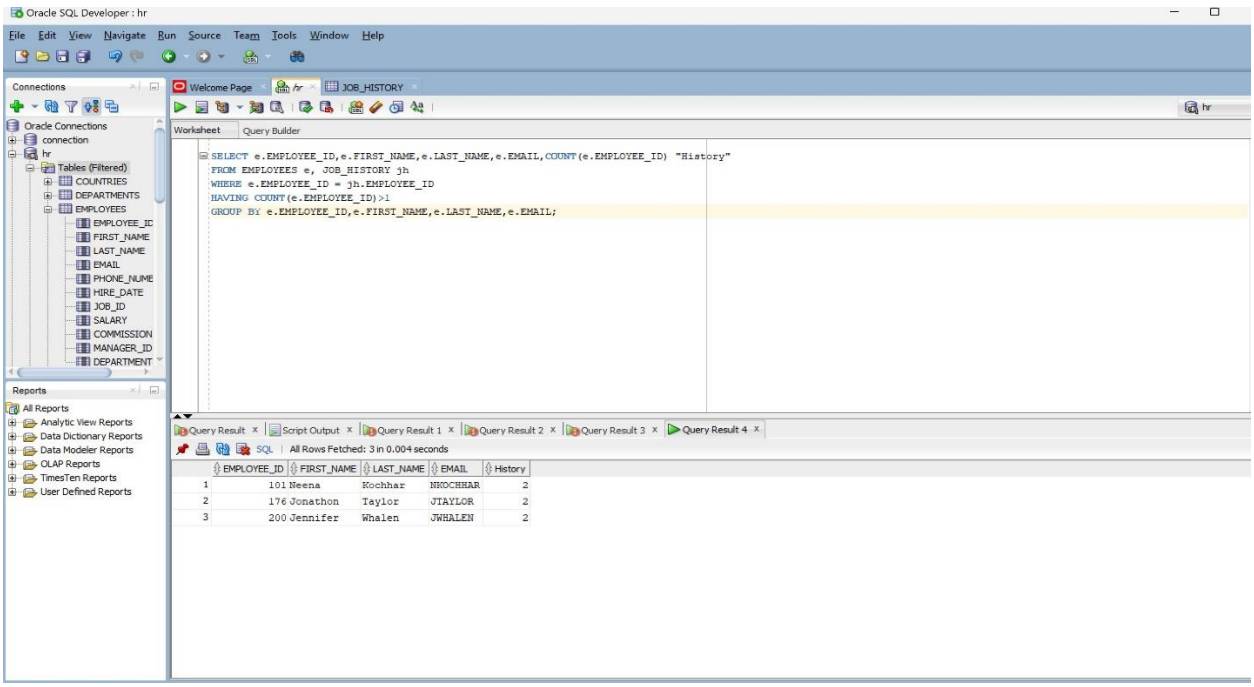
SOLUTION:

a)
```
SELECT EMPLOYEE_ID, COUNT(*) AS “Number_Of_Jobs”
FROM JOB_HISTORY
GROUP BY EMPLOYEE_ID
ORDER BY COUNT(*) DESC;
```



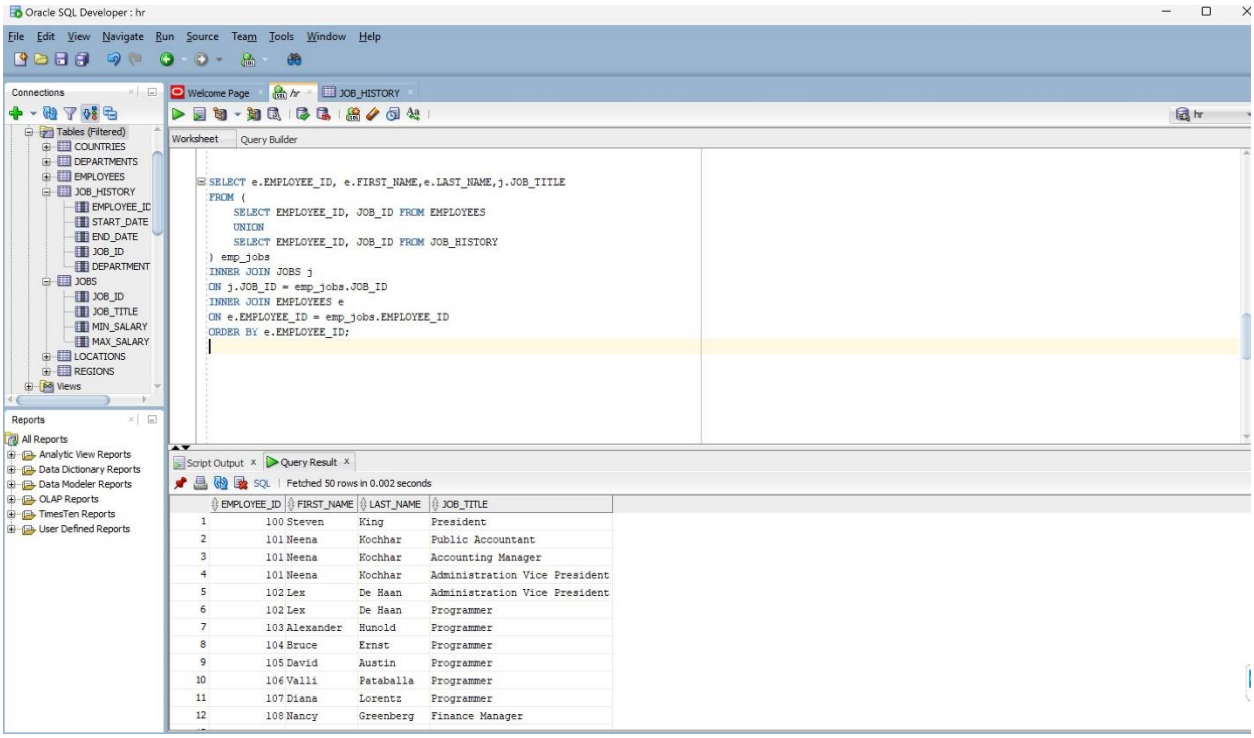
b)

```
SELECT e.EMPLOYEE_ID, e.FIRST_NAME, e.LAST_NAME, e.EMAIL,
COUNT(e.EMPLOYEE_ID) “History”
FROM EMPLOYEE e, JOB_HISTORY jh
WHERE e.EMPLOYEE_ID = jh.EMPLOYEE_ID
HAVING COUNT (e.EMPLOYEE_ID) >1
GROUP BY e.EMPLOYEE_ID, e.FIRST_NAME, e.LAST_NAME, e.EMAIL;
```



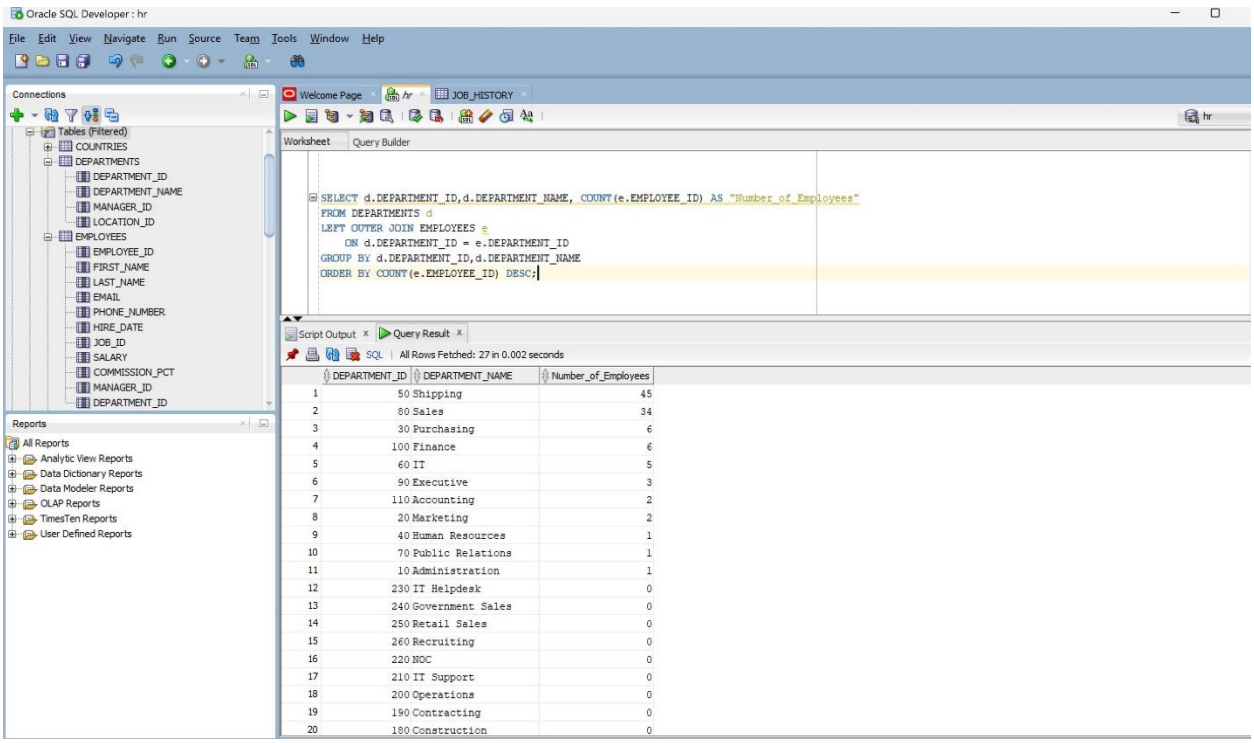
c)

```
SELECT e.EMPLOYEE_ID, e.FIRST_NAME, e.LAST_NAME, JOB_TITLE
FROM (
        SELECT EMPLOYEE_ID, JOB_ID FROM EMPLOYEES
        UNION
        SELECT EMPLOYEE_ID, JOB_ID FROM JOB_HISTORY
) emp_jobs
INNER JOIN JOBS j
ON j.JOB_ID = emp_jobs.JOB_ID
INNER JOIN EMPLOYEE e
ON e. EMPLOYEE_ID = emp_jobs.EMPLOYEE_ID
ORDER BY e.EMPLOYEE_ID;
```



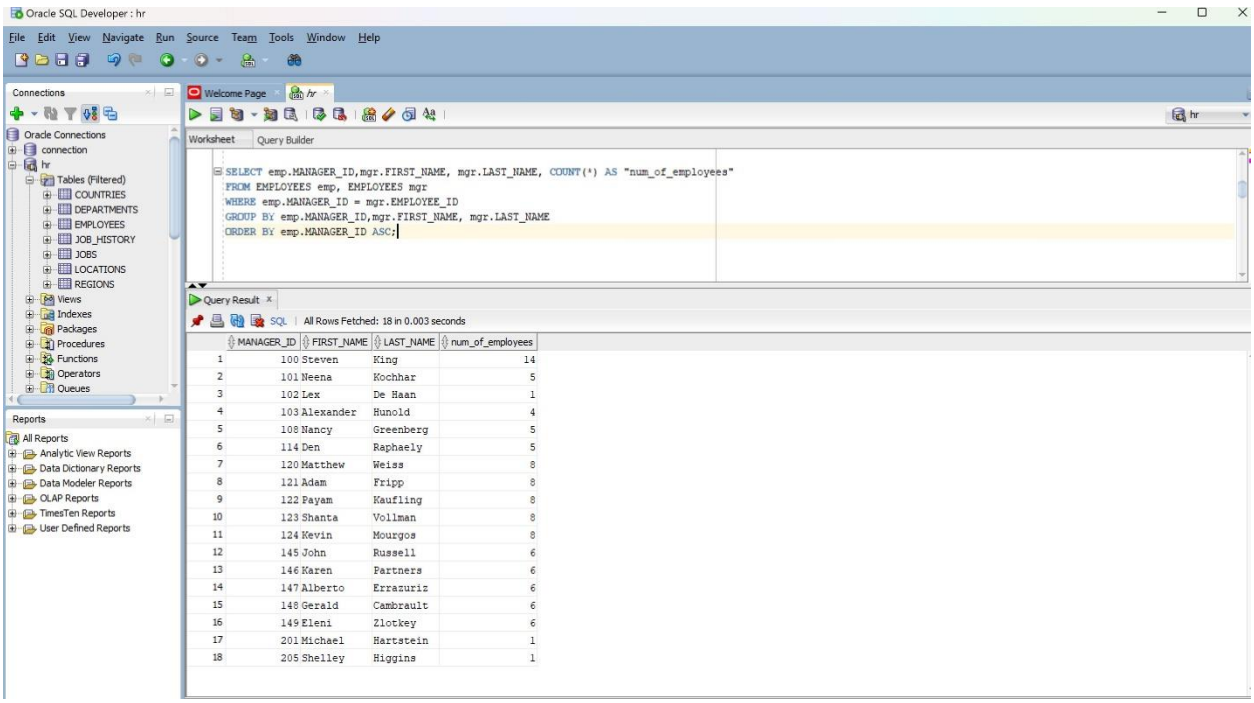
d)

```
SELECT d.DEPARTMENT_ID, d.DEPARTMENT_NAME,
COUNT(e.EMPLOYEE_ID) AS “Number_Of_Employees”
FROM DEPARTMENTS d
LEFT OUTER JOIN EMPLOYEES e
        ON d. DEPARTMENT_ID, d.DEPARTMENT_NAME
ORDER BY COUNT(e.EMPLOYEE_ID) DESC;
```



e)

SELECT emp.MANAGER_ID mgr.FIRST_NAME, mgr.LAST_NAME , COUNT(*) AS “num_of_employees”
FROM EMPLOYEES emp, EMPLOYEES mgr
WHERE emp.MANAGER_ID = mgr.EMPLOYEE_ID
GROUP BY emp.MANAGER_ID, mgr.FIRST_NAME, mgr.LAST_NAME
ORDER BY emp.MANAGER-ID ASC;



Feedback and grade:
97 marks

HW7

3. d. -3 Incorrect syntax. The asterisk symbol (*) should not be included in the DELETE statement. The EXCEPT operator is not supported in Oracle. Instead, you should use the MINUS operator to achieve the same result.