CN5000 Tutorials

Jayrup Nakawala

Table of contents

1 Week 1

1.1 Compare and contrast between the main characteristics of the database and file-based approach.

- **File-based Approach**: A file based system is a collection of applications which uses their own, seperate databases to manage their data. Resulting in data redundancy and lower reliability. Although it is easier to set up. This apporach allows data to be stored in a structured or unstructured way, eg, csv, json, docx, xlss.

- **Database Apporach**: A database management system, creates a system for the collection of application which allows them to store data centrally. This approach allows the user to create relationships between data through primary and foreign keys and helps increase data integrity through constraints and views. This approach allows data to be stored only in a structured way.

1.2 Describe four government sectors that would be expected to make use of database systems.

- **Sector for Transport Management** – for traffic managemnet, drivers and vehicle records, transport planning, etc.
- **Sector for Education – national** pupil database, exams and assessment records, school census results, etc.
- **Sector for Business & Trade** – business registration records, supply chain management, international trade, etc.

- **Sector for Health & Social Care** – patient records, public health databases, social care, etc.

1.3 What is meant by the terms:

- **Database**: In simplest terms, it is a collection of structured related data and its descriptions. Which is designed to meet the information needs of an organization.
- **Database Management System**: A management system which allows the user to interact with the databse. Which includes the ability to create, define, maintain and control access to the database via the Data Manipulation Language. It serves as an intermidiatory between the database and the application (or users). Examples: MySQL, MongoDB, PostgreSQL, etc.
- **Database Application Program**: It is an application designed to interact with a database in a more user-friendly interface making the database interactions easier for non-technical users. Some common examples include: Microsoft Access, Oracle forms, etc.
- **Data Independence**: Data independece refers to the way a DBMS seperates the data description from the application, thereby making the applications immune to changes in data descriptions.
- **Security**: Database security is the protection of the database by constricting access to it. For example, the Database Admin can access all the data in the database but, lets say, a manager of the marketing team can only access data refering to their field. The operation allowed can also be restricted, for example, an intern may only be able to view a small subset of data and not be allowed to make any changes to it.
- **Integrity**: Database integrity means the validity and consistancy of the datastored which can be expressed in terms of constrains. Constraints are way of defining rules between a single record or to relationships between records.
- **Views**: A view is a subset of the database. It allows for better security by only let the people who have acces view certain data.

1.4 Describe the role of database management systems (DBMS) in the database approach. Discuss why knowledge of DBMS is important for database administrators.

DBMS are curcial to the database approach. This approach involves, rganizing, storing, and managing data in a way that is independent of the application programs that use the data. Database Administrators are responsible for ensuring that databases operate efficiently, securely, and without disruption. Knowledge of how DBMS works makes them more effiecent at their work and allows them to perform taskes much quicky all the while increasing securing and integrity of the dat

1.5 Describe the five components of the DBMS environment.

-**Hardware**: It refers to the hardware the database runs on. Including, the servers, which runs the DBMS, the switchs which enable networnking and fans which allows for cooling of the systems.

-**Software**: it refers to the actual DBMS software which handles the data, as well as the OS and any networking software.

-**Data**: Perhaps the most important part of the DBMS environment. It is the main reason DBMS exists and it also acts as a bridge between the human component and the machine component of the DBMS environment.

-**Procedures**: These are the rules and insurctions which govern the design and use of the database. Including instructions on how to log in, use a specific functionality, start and stop the DBMS, making backups, etc.

-**People**: There are 4 distinct kind of people who participate in the DBMS environment, namely, data and database administartors, database designers, application developers, and end-users.

1.6 What are the advantages of database management systems? Are there any potential disadvantages?

Here are the potential advantages and disadvantages of the database management system:

Advantages	Disadvantages
Control of data redundancy	Complexity
Data consistancy	Size of DMBS
More info for the same amount of data	Cost of DBMS
Sharing of data	Cost of conversion
Improved data security	Performance
Enforcement of standards	Additional hardware costs
Improved data integrity	Greater impact of a failure

1.7 Discuss the roles of the following personnel in the database environment:

- **Data Administrator**: is responsible for the management of the data resource, including database planning; development and maintenance of standards, policies and procedures; and conceptual/logical database design.

- **Database Administrator**: is responsible for the physical realization of the database, including physical database design and implementation, security and integrity control, maintenance of the operational system, and ensuring satisfactory performance of the applications for users.
- **Logical Database Designer**: The logical database designer is concerned with identifying the data (that is, the entities and attributes), the relationships between the data, and the constraints on the data that is to be stored in the database.
- **Physical Database Designer**: The physical database designer decides how the logical database design is to be physically realized, including the mapping of logical designs to the sets of tables and constraints, selecting specifc storage structure, and security.
- **Application Developer**: Creates the application program that provides the required functionality for the end-users.
- **End-Users**: The end-users are the "clients" of the database, which has been designed and implemented and is being maintained to serve their information needs. They can be novice or sophisticated.

1.8 Discuss the three generations of DBMSs.

The history of Database Management Systems (DBMSs) can be divided into three generations, each representing significant technological advancements and addressing the limitations of the previous generation:

- First-Generation DBMSs: Hierarchical and Network Models
- These ranged from Mid-1960s to early 1970s and had tree based hierarchical structure or complex network models which were cumbersome to maintain and expand.
- Second-Generation DBMSs: Relational Model
- Ranges from 1970s to 1980s, and introduced a fundamentally different relational model which improved data independence.
- Third-Generation DBMSs: Object-Oriented and Object-Relational Models
- From 1990s to present, this generation gave a more object oriented approach allowing to overcome the limitations of the previous generations. This means support for inheritance, encapsulation, and polymorphism and more powerfull and efficient tools.

1.9 Why are views an important aspect of database management systems?

A view is, in essence, some subset of the database. It is important for the following reasons:

- Views provide a level of security.Views can be set up to exclude data that some users should not see.
- Views provide a mechanism to customize the appearance of the database.
- A view can present a consistent, unchanging picture of the structure of the database, even if the underlying database is changed

2 Week 2

2.1 Describe the concept of database schema and explain the three types of schema in a database.

For data independence, it is important to keep the database and the description of the database seperate. This description of the database is called the database schema. There are three different types of schema based on their level of abstraction.

- External Schema aka, subschema. These are the highest level of abstraction which relates to the different views of the database. It describes the part of the database which is relevent to the user.
- Conceptual Schema This describes what data is stored and its relationships between them, which includes all the entities, attributes and relationships together with integrity constraints.
- Internal Schema This is the lowest level of abstraction.

2.2 What is a data model? What are the three main components of a data model?

A Data Model in Database is a model describing how the real-world objects are organized and how these are associated within the database in a visual representation. It is an abstraction that provides understanding of the database design to the end-users and database designers. It summarizes the description of the database helping to implement the design developed.

Three main components of the data model are: - Structural part that contains the set of rules which the database is structured or built. - Manipulative part consitiutes what commands can be used to add, delete or alter the data present in the database. - Integrity constraints that maintain data consistency.

2.3 What functions and services would you expect a multi-user DBMS provide?

A multi-user DBMS shoukd have the following functions and services.

- concurrency
- network connection management
- providing access to all the database servers
- database connection pooling
- legacy database support
- clustering support
- load balancing
- failover

2.4 Which functions and services would not be required for a standalone PC DBMS?

Following are the functions not required for standalone PC DMS:

- Multi-user Access: Since it is a standalone PC DBMS, multi-user access function is not required.
- Network communication: Standalone PC DBMS don't communicate with other computers; network protocols are unnecessary.
- Concurrency Control System: Concurrency Control is needed when multiple users are using and updating the same database simultaneously. Since standalone PC DBMS is accessed by only one user, concurrency control service is not essential.

2.5 Describe the function a system catalogue. What are the benefits of having a system catalog?

Any database management system that describes the database objects has a system catalogue at its core. It includes details on the physical and logical construction of database objects, including packages, tables, views, indexes, and statistical data. System catalogue provides vast information about your database management system. The benefits of having a system catalog include centralizing data, simplifying communications, discovering, and regulating redundancies and inconsistencies, and assuring security and integrity.

2.6 What is the difference between DDL and DML?

- DDL (Data Definition Language): Defines or modifies database structure (e.g., CREATE, ALTER, DROP).
  - Example:

```
    CREATE TABLE Students (ID INT, Name VARCHAR(50));
```

- DML (Data Manipulation Language): Manipulates data in the database (e.g., INSERT, UPDATE, DELETE, SELECT).
  - Example:

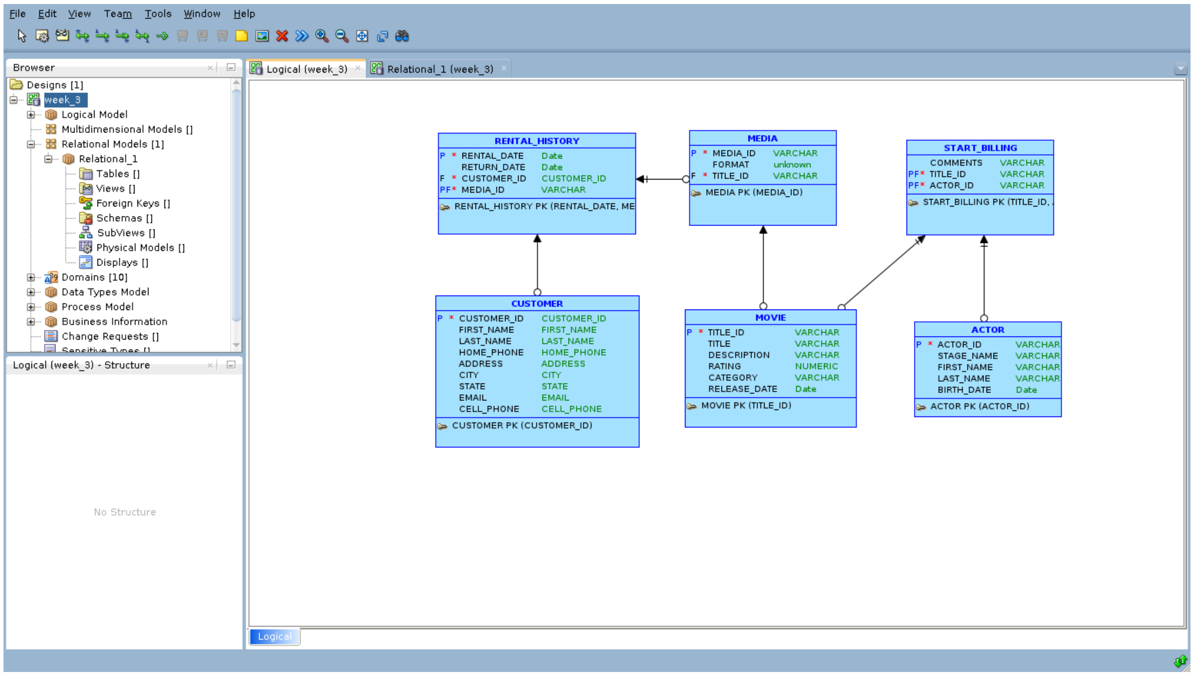
```
  INSERT INTO Students (ID, Name) VALUES (1, 'Alice');
```

2.7 Name three record-based data models. Discuss the main differences between these data models.

Three record-based data models are as follows:
- Hierarchical model: Organizes data in tree-like structure representing data in parent-child hierarchy. - Network model: Like Hierarchical model Organizes data in graph-like structure representing more complex relationships. - Relational model: Organizes data in tables – rows and columns. Tables are associated with each other using foreign keys.

3 Week 3

3.1 Capture the screenshots of your solution, using snipping tool and put them into your document.



```
RENTAL_HISTORY
P  * RENTAL_DATE     Date
     RETURN_DATE     Date
F  * CUSTOMER_ID     CUSTOMER_ID
PF * MEDIA_ID        VARCHAR
RENTAL_HISTORY PK (RENTAL_DATE, ME

MEDIA
P  * MEDIA_ID        VARCHAR
     FORMAT          unknown
F  * TITLE_ID        VARCHAR
MEDIA PK (MEDIA_ID)

START_BILLING
     COMMENTS        VARCHAR
PF * TITLE_ID        VARCHAR
PF * ACTOR_ID        VARCHAR
START_BILLING PK (TITLE_ID, .

CUSTOMER
P  * CUSTOMER_ID     CUSTOMER_ID
     FIRST_NAME      FIRST_NAME
     LAST_NAME       LAST_NAME
     HOME_PHONE      HOME_PHONE
     ADDRESS         ADDRESS
     CITY            CITY
     STATE           STATE
     EMAIL           EMAIL
     CELL_PHONE      CELL_PHONE
CUSTOMER PK (CUSTOMER_ID)

MOVIE
P  * TITLE_ID        VARCHAR
     TITLE           VARCHAR
     DESCRIPTION     VARCHAR
     RATING          NUMERIC
     CATEGORY        VARCHAR
     RELEASE_DATE    Date
MOVIE PK (TITLE_ID)

ACTOR
P  * ACTOR_ID        VARCHAR
     STAGE_NAME      VARCHAR
     FIRST_NAME      VARCHAR
     LAST_NAME       VARCHAR
     BIRTH_DATE      Date
ACTOR PK (ACTOR_ID)
```

Figure 1: erd

3.2 What are the main stages of the database system development lifecycle? Depict the stages using *www.draw.io* . For each stage, state if it is mandatory or optional.

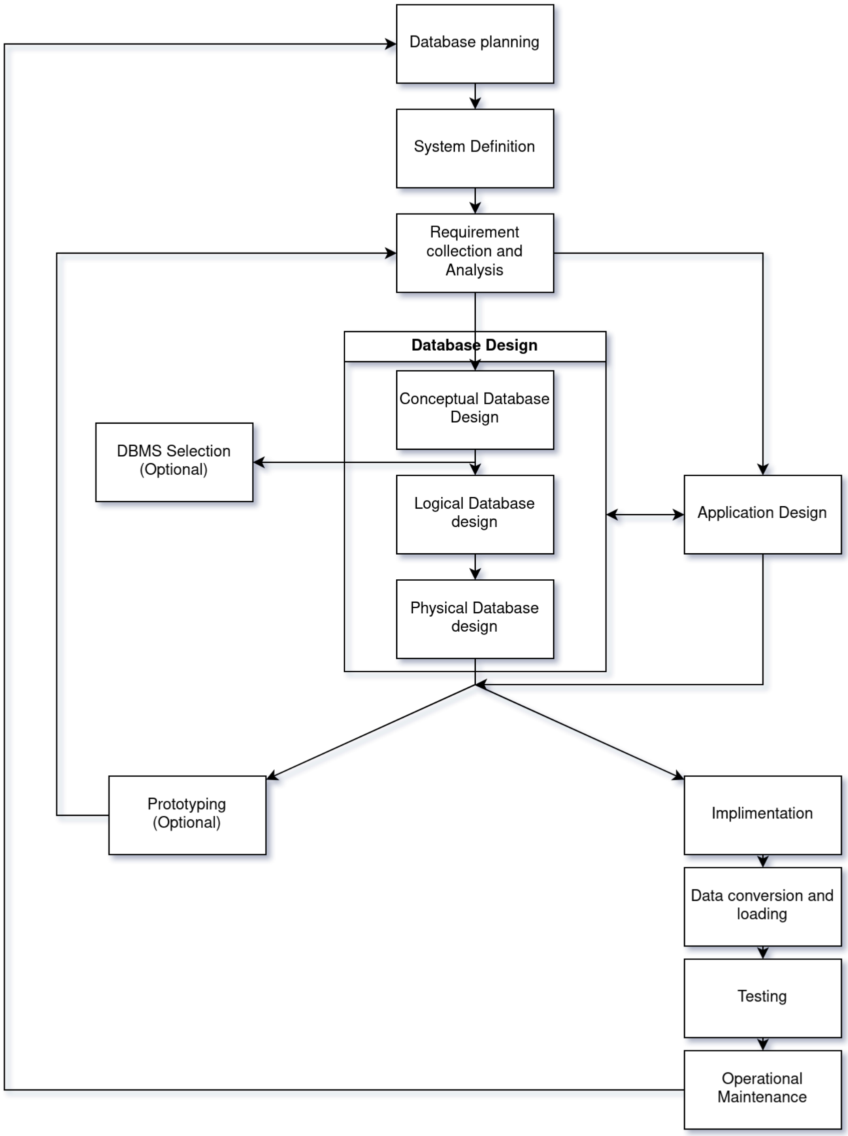


Figure 2: flowchart

Main Stages of the Database Development Lifecycle:

- Database Planning Lays the foundation for all other stages of the project. (Mandatory)

- System Definition Defines the goals and scope of the project.
- Requirement Collection and Analysis Documents specific needs and assesses project feasibility.
- Database Design Builds the data architecture, including:
  - Conceptual Design
  - Logical Design
  - Physical Design
- DBMS Selection (Optional) Selects a Database Management System (DBMS) that meets project requirements.
- Application Design Plans the use of application programs and user interfaces.
- Prototyping (Optional) Creates a working model of the database.
- Implementation Bridges the gap between idea and execution. (Mandatory)
- Data Conversion and Loading Transfers and uploads existing data into the new system.
- Testing Ensures the system is validated for data integrity through various tests. (Mandatory)

3.3 Discuss the main approaches to database design. Discuss the contexts where each is appropriate.

The main approaches to database design are:

- **Bottom-Up Approach**: This approach starts with the smallest data elements and focuses on their relationships. These elements are gradually combined to form larger entities and structures.
  - Appropriate Context: Ideal for systems where data elements are well-defined, such as legacy systems or projects with detailed existing data but no clear overall structure.
- **Top-Down Approach**: The design begins at a high level by identifying key entities, their attributes, and the relationships between them. The entities are then broken down further to add details.
  - Appropriate Context: Suitable for new systems where an overall structure is clear from the start but specific details will emerge later, like enterprise-level applications.
- **Inside-Out Approach**: This method focuses on defining the most critical or central entities first. The design is then expanded to incorporate related entities and attributes.
  - Appropriate Context: Best for systems where a core functionality or central component is already known, such as modular or specialized applications.
- **Mixed Strategy**: A hybrid approach that combines top-down and bottom-up methods. High-level design is applied to some parts of the system, while detailed data structures are addressed in others.

  - Appropriate Context: Useful for complex systems that require flexibility, such as large-scale projects with varying levels of detail across different modules.

3.4 What are the three phases of database design? How are they related to each other?

Conceptual Design

- Defines high-level data requirements using models like Entity-Relationship Diagrams (ERDs).
- Focuses on identifying entities, relationships, and constraints without implementation details.

Logical Design

- Maps the conceptual model into a logical schema compatible with the chosen DBMS.
- Defines tables, attributes, keys, and relationships.

Physical Design

- Implements the logical design by optimizing storage structures, indexing, and access paths for performance.

Relationship : Each phase builds on the previous one: the conceptual design captures what data is needed, the logical design structures it for the DBMS, and the physical design implements and optimizes it for storage and performance.

3.5 The following are problems that have been identified during the testing process in the development of a new system. In which part of the life cycle do you think these problem could have originated and been identified by a thorough review following that stage in the development life cycle?

The performance of the system is poor – failing to respond quickly enough to meet the stated user requirement of interactive, screen-based use.

- Physical Design Phase: The system's performance issues likely stem from poor optimization of storage structures, indexing, or access paths. A thorough review here could have addressed performance requirements.

No backup facilities were included to meet the users' requirement of long-term archival of their data.

- Requirements Collection and Analysis Phase: Backup requirements were likely missed during this phase. Proper documentation and review would have ensured backup and archival needs were addressed.

No user manuals were provided!

- Implementation or Testing Phase: User documentation was overlooked or deprioritized. A thorough review during these stages could have identified and resolved the issue before deployment.

4 week 4

4.1 For the case-study design of Company, design the full (Advance) E-R diagram that shows the link between them and list all the attributes along with PK and FK using oracle data modeler.

- COMPANY
  - Employees, departments, and projects
  - Company is organized into departments
  - Department controls a number of projects
  - Employee: store each employee's name, Social Security number, address, salary, sex (gender), and birth date
  - Keep track of the dependents of each employe

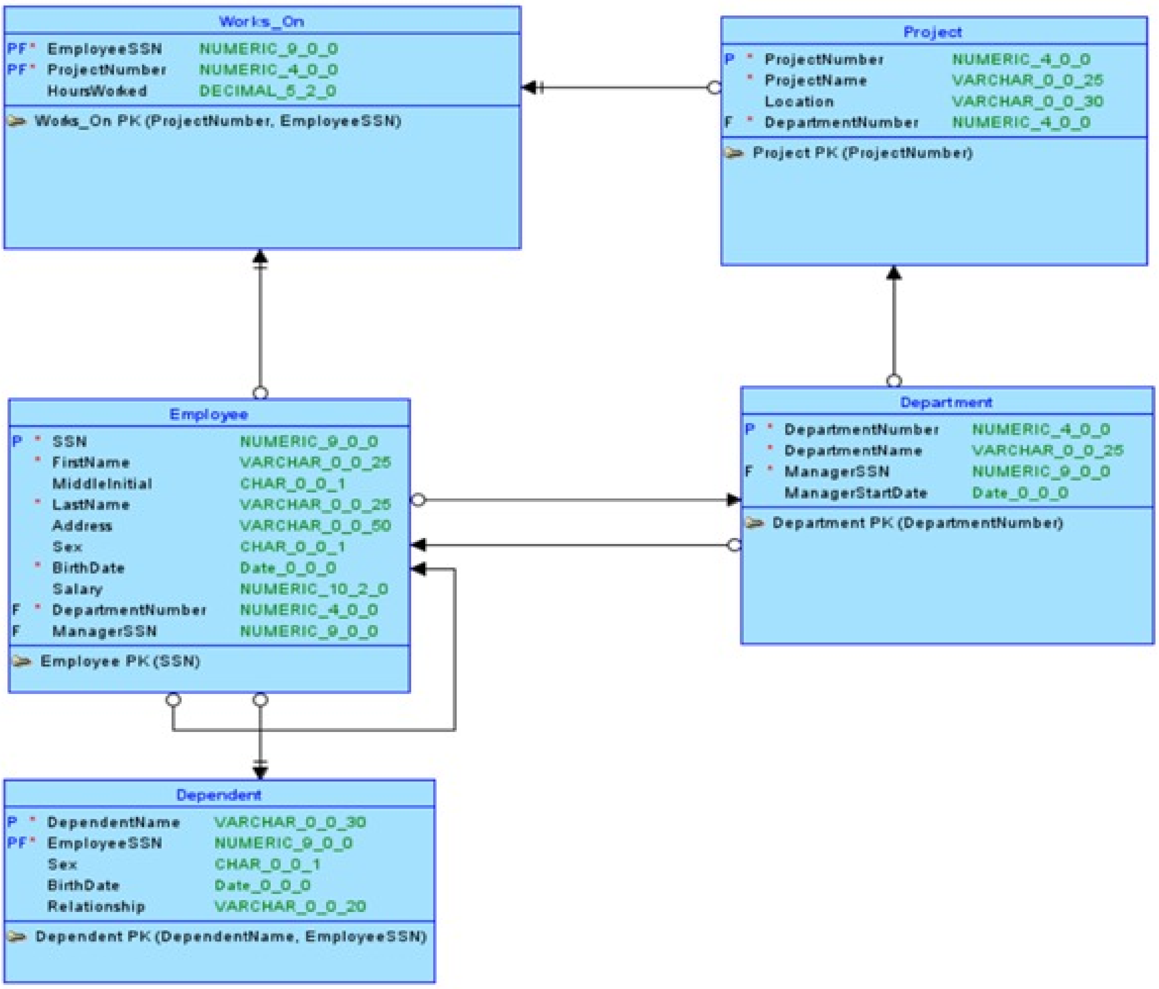

Works_On
PF* EmployeeSSN NUMERIC_9_0_0
PF* ProjectNumber NUMERIC_4_0_0
HoursWorked DECIMAL_5_2_0
Works_On PK (ProjectNumber, EmployeeSSN)
Project
P * ProjectNumber NUMERIC_4_0_0
* ProjectName VARCHAR_0_0_25
Location VARCHAR_0_0_30
F * DepartmentNumber NUMERIC_4_0_0
Project PK (ProjectNumber)
Employee
P * SSN NUMERIC_9_0_0
* FirstName VARCHAR_0_0_25
MiddleInitial CHAR_0_0_1
* LastName VARCHAR_0_0_25
Address VARCHAR_0_0_50
Sex CHAR_0_0_1
* BirthDate Date_0_0_0
Salary NUMERIC_10_2_0
F * DepartmentNumber NUMERIC_4_0_0
F ManagerSSN NUMERIC_9_0_0
Employee PK (SSN)
Department
P * DepartmentNumber NUMERIC_4_0_0
* DepartmentName VARCHAR_0_0_25
F * ManagerSSN NUMERIC_9_0_0
ManagerStartDate Date_0_0_0
Department PK (DepartmentNumber)
Dependent
P * DependentName VARCHAR_0_0_30
PF* EmployeeSSN NUMERIC_9_0_0
Sex CHAR_0_0_1
BirthDate Date_0_0_0
Relationship VARCHAR_0_0_20
Dependent PK (DependentName, EmployeeSSN)

Figure 3: erd

4.2 Which normal form (Normalization) do the attributes in these entities belong to ?

1 NF

4.3 What will be the 0NF attributes of this system ?

As the data is already in 1NF form, there will be no 0NF attributes.

4.4 Draw a simple usecase for the above system.

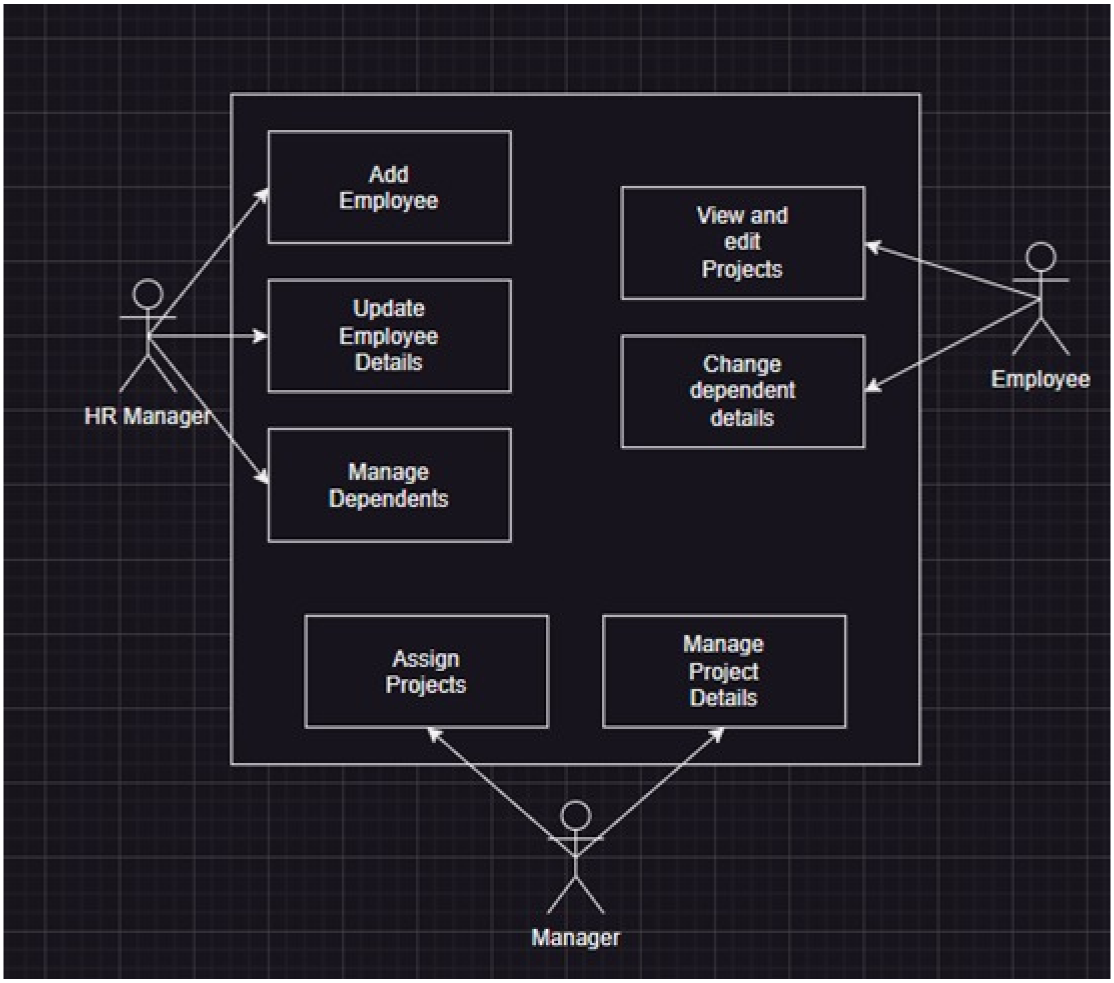
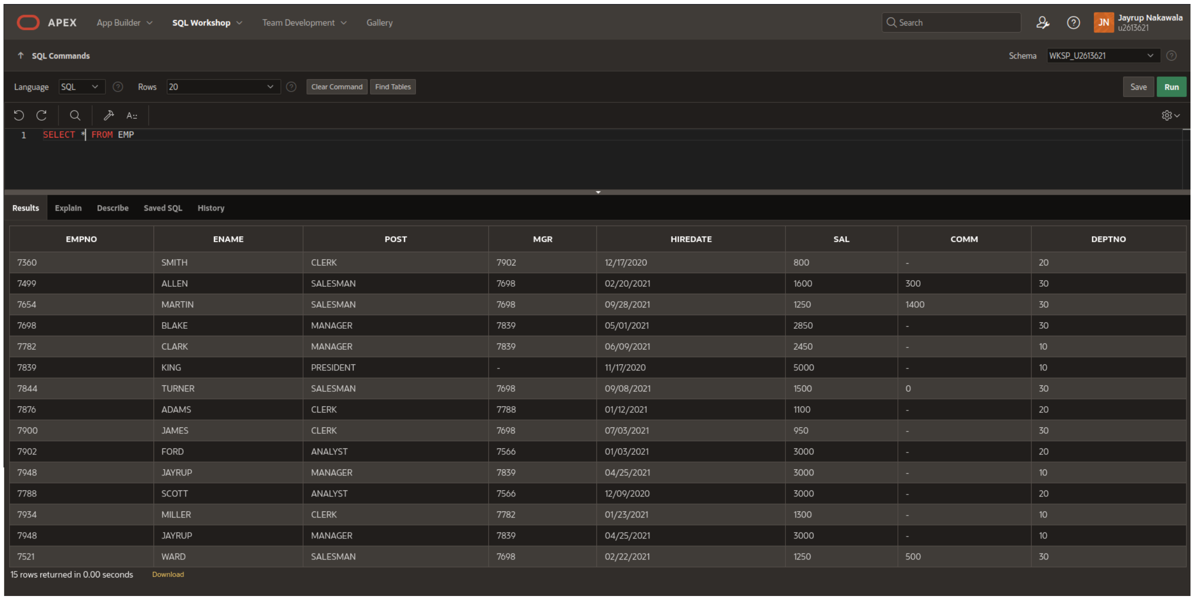


Figure 4: usecase

5 Week 5

5.1 Output for SQL Command:

```
SELECT * FROM EMP
```



Schema WKSP_U2613621

Language SQL Rows 20

```
SELECT * FROM EMP
```

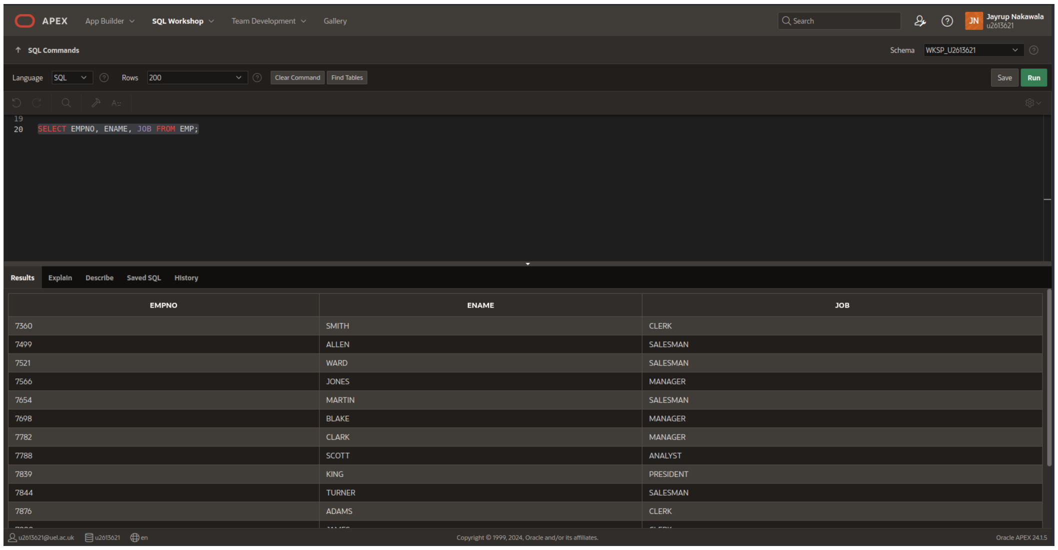
EMPNO	ENAME	POST	MGR	HIREDATE	SAL	COMM	DEPTNO
7360	SMITH	CLERK	7902	12/17/2020	800	-	20
7499	ALLEN	SALESMAN	7698	02/20/2021	1600	300	30
7654	MARTIN	SALESMAN	7698	09/28/2021	1250	1400	30
7698	BLAKE	MANAGER	7839	05/01/2021	2850	-	30
7782	CLARK	MANAGER	7839	06/09/2021	2450	-	10
7839	KING	PRESIDENT	-	11/17/2020	5000	-	10
7844	TURNER	SALESMAN	7698	09/08/2021	1500	0	30
7876	ADAMS	CLERK	7788	01/12/2021	1100	-	20
7900	JAMES	CLERK	7698	07/03/2021	950	-	30
7902	FORD	ANALYST	7566	01/03/2021	3000	-	20
7948	JAYRUP	MANAGER	7839	04/25/2021	3000	-	10
7788	SCOTT	ANALYST	7566	12/09/2020	3000	-	20
7934	MILLER	CLERK	7782	01/23/2021	1300	-	10
7948	JAYRUP	MANAGER	7839	04/25/2021	3000	-	10
7521	WARD	SALESMAN	7698	02/22/2021	1250	500	30

15 rows returned in 0.00 seconds

Figure 5: table1

5.2 Output for SQL Command:

```
SELECT EMPNO, ENAME, JOB_NAME FROM EMP
```



Schema WKSP_U2613621

Language SQL Rows 200

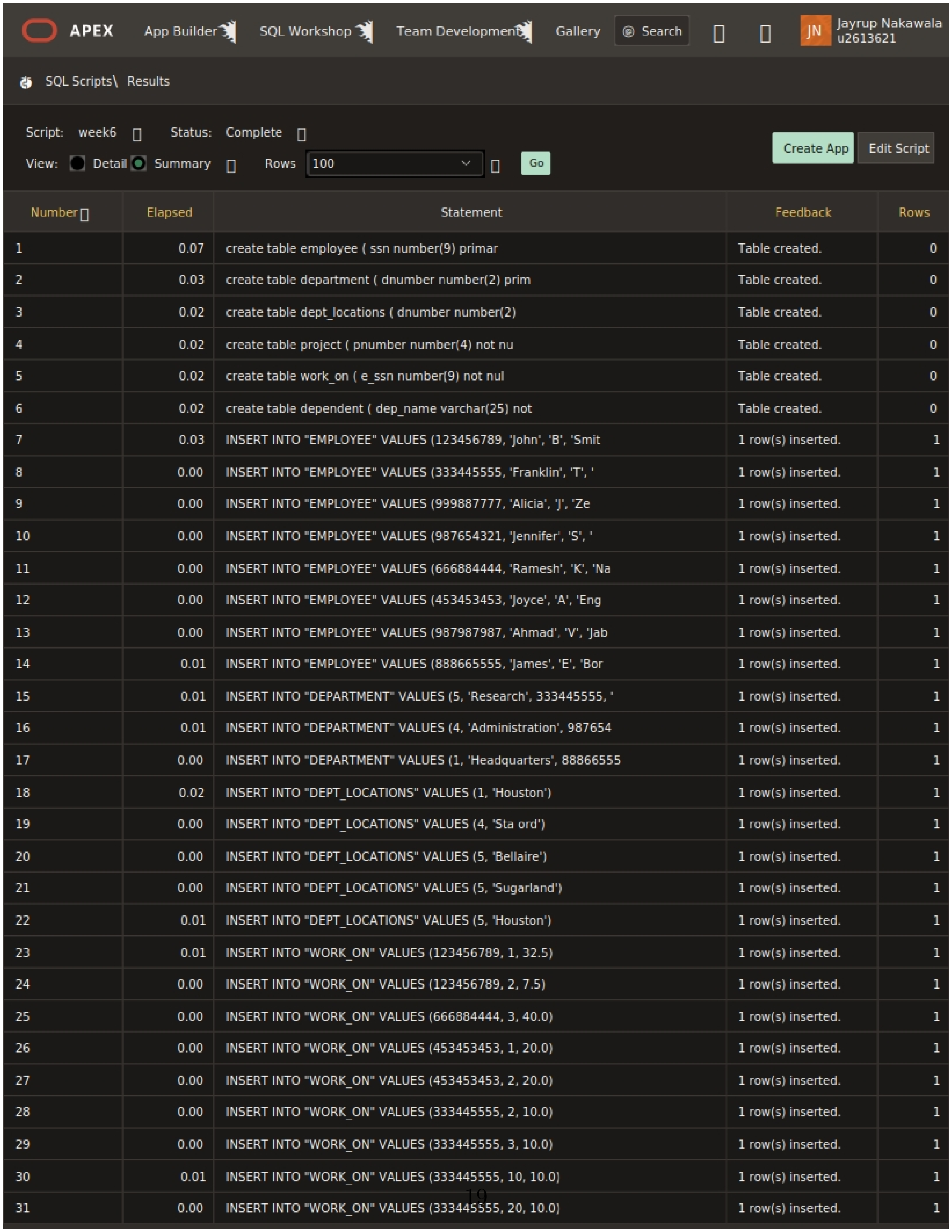
```
SELECT EMPNO, ENAME, JOB FROM EMP;
```

EMPNO	ENAME	JOB
7360	SMITH	CLERK
7499	ALLEN	SALESMAN
7521	WARD	SALESMAN
7566	JONES	MANAGER
7654	MARTIN	SALESMAN
7698	BLAKE	MANAGER
7782	CLARK	MANAGER
7788	SCOTT	ANALYST
7839	KING	PRESIDENT
7844	TURNER	SALESMAN
7876	ADAMS	CLERK

Figure 6: table2

6 Week 6

6.1 Create tables and insert values in them based on the given erd.

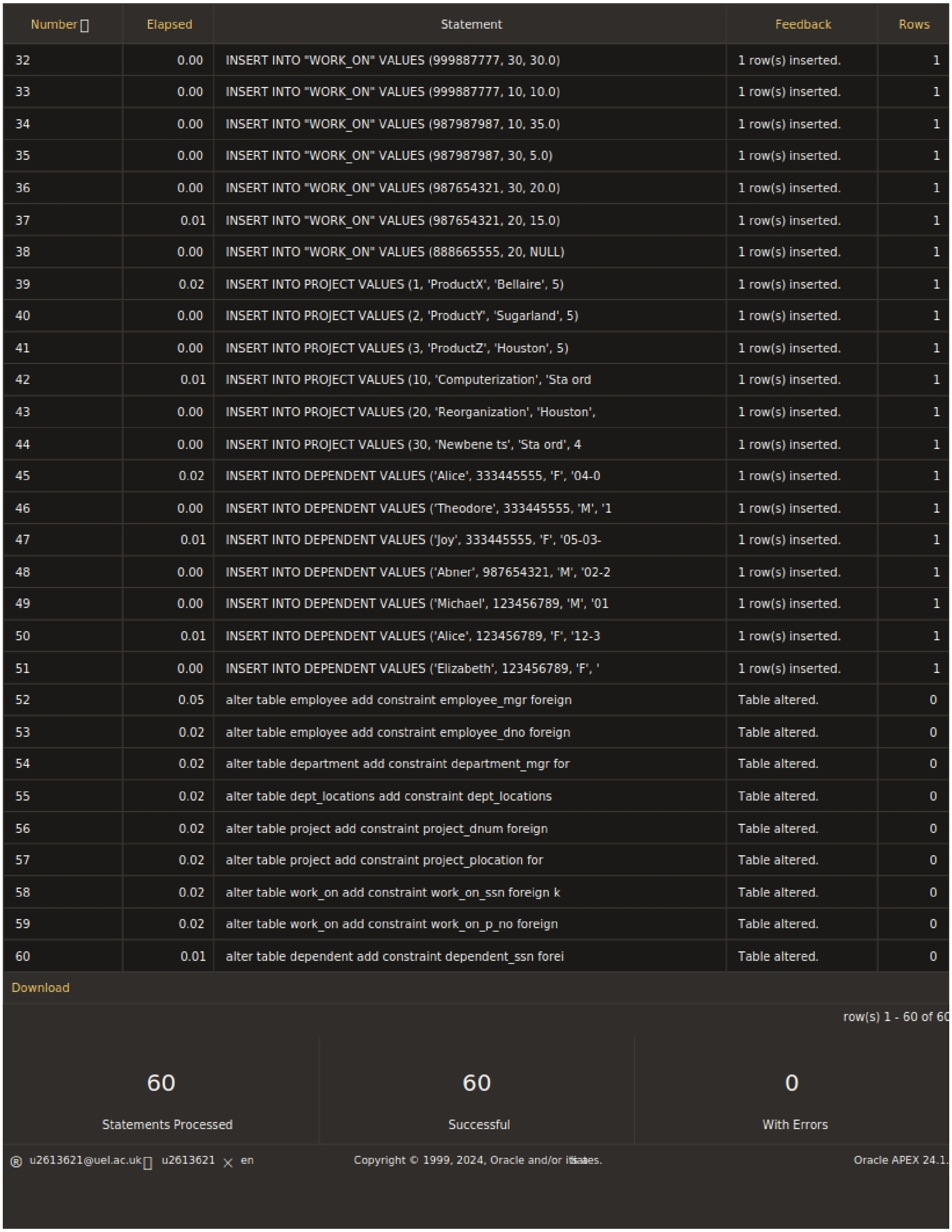


Script: week6 Status: Complete

View: ● Detail ◉ Summary Rows 100

Number	Elapsed	Statement	Feedback	Rows
1	0.07	create table employee (ssn number(9) primar	Table created.	0
2	0.03	create table department (dnumber number(2) prim	Table created.	0
3	0.02	create table dept_locations (dnumber number(2)	Table created.	0
4	0.02	create table project (pnumber number(4) not nu	Table created.	0
5	0.02	create table work_on (e_ssn number(9) not nul	Table created.	0
6	0.02	create table dependent (dep_name varchar(25) not	Table created.	0
7	0.03	INSERT INTO "EMPLOYEE" VALUES (123456789, 'John', 'B', 'Smit	1 row(s) inserted.	1
8	0.00	INSERT INTO "EMPLOYEE" VALUES (333445555, 'Franklin', 'T', '	1 row(s) inserted.	1
9	0.00	INSERT INTO "EMPLOYEE" VALUES (999887777, 'Alicia', 'J', 'Ze	1 row(s) inserted.	1
10	0.00	INSERT INTO "EMPLOYEE" VALUES (987654321, 'Jennifer', 'S', '	1 row(s) inserted.	1
11	0.00	INSERT INTO "EMPLOYEE" VALUES (666884444, 'Ramesh', 'K', 'Na	1 row(s) inserted.	1
12	0.00	INSERT INTO "EMPLOYEE" VALUES (453453453, 'Joyce', 'A', 'Eng	1 row(s) inserted.	1
13	0.00	INSERT INTO "EMPLOYEE" VALUES (987987987, 'Ahmad', 'V', 'Jab	1 row(s) inserted.	1
14	0.01	INSERT INTO "EMPLOYEE" VALUES (888665555, 'James', 'E', 'Bor	1 row(s) inserted.	1
15	0.01	INSERT INTO "DEPARTMENT" VALUES (5, 'Research', 333445555, '	1 row(s) inserted.	1
16	0.01	INSERT INTO "DEPARTMENT" VALUES (4, 'Administration', 987654	1 row(s) inserted.	1
17	0.00	INSERT INTO "DEPARTMENT" VALUES (1, 'Headquarters', 88866555	1 row(s) inserted.	1
18	0.02	INSERT INTO "DEPT_LOCATIONS" VALUES (1, 'Houston')	1 row(s) inserted.	1
19	0.00	INSERT INTO "DEPT_LOCATIONS" VALUES (4, 'Sta ord')	1 row(s) inserted.	1
20	0.00	INSERT INTO "DEPT_LOCATIONS" VALUES (5, 'Bellaire')	1 row(s) inserted.	1
21	0.00	INSERT INTO "DEPT_LOCATIONS" VALUES (5, 'Sugarland')	1 row(s) inserted.	1
22	0.01	INSERT INTO "DEPT_LOCATIONS" VALUES (5, 'Houston')	1 row(s) inserted.	1
23	0.01	INSERT INTO "WORK_ON" VALUES (123456789, 1, 32.5)	1 row(s) inserted.	1
24	0.00	INSERT INTO "WORK_ON" VALUES (123456789, 2, 7.5)	1 row(s) inserted.	1
25	0.00	INSERT INTO "WORK_ON" VALUES (666884444, 3, 40.0)	1 row(s) inserted.	1
26	0.00	INSERT INTO "WORK_ON" VALUES (453453453, 1, 20.0)	1 row(s) inserted.	1
27	0.00	INSERT INTO "WORK_ON" VALUES (453453453, 2, 20.0)	1 row(s) inserted.	1
28	0.00	INSERT INTO "WORK_ON" VALUES (333445555, 2, 10.0)	1 row(s) inserted.	1
29	0.00	INSERT INTO "WORK_ON" VALUES (333445555, 3, 10.0)	1 row(s) inserted.	1
30	0.01	INSERT INTO "WORK_ON" VALUES (333445555, 10, 10.0)	1 row(s) inserted.	1
31	0.00	INSERT INTO "WORK_ON" VALUES (333445555, 20, 10.0)	1 row(s) inserted.	1

Figure 7: table1



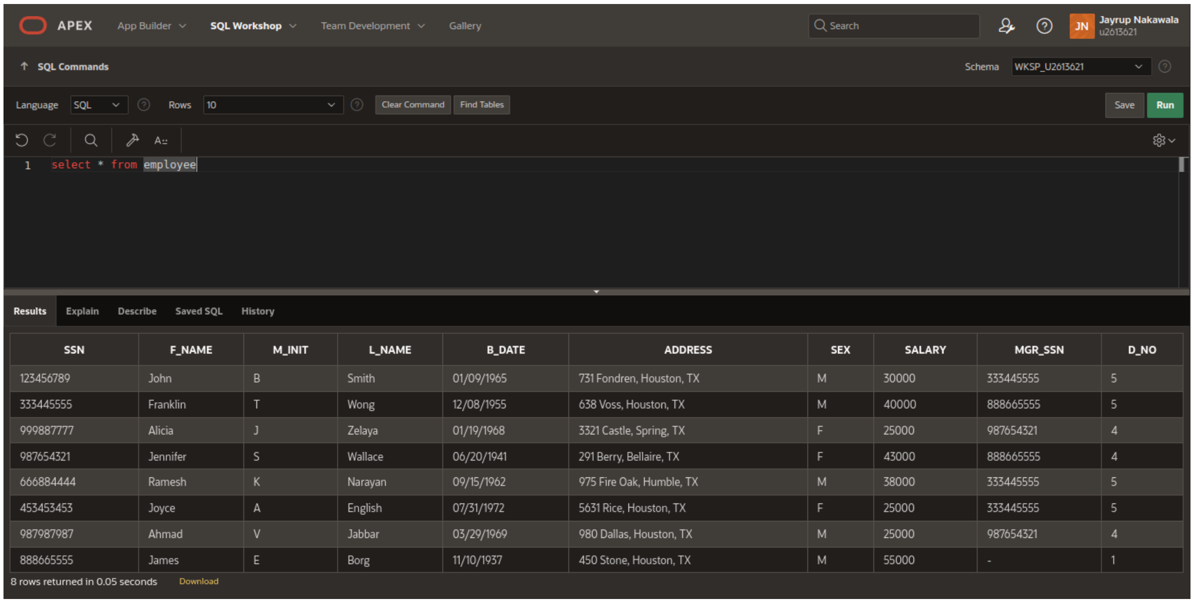
Number	Elapsed	Statement	Feedback	Rows
32	0.00	INSERT INTO "WORK_ON" VALUES (999887777, 30, 30.0)	1 row(s) inserted.	1
33	0.00	INSERT INTO "WORK_ON" VALUES (999887777, 10, 10.0)	1 row(s) inserted.	1
34	0.00	INSERT INTO "WORK_ON" VALUES (987987987, 10, 35.0)	1 row(s) inserted.	1
35	0.00	INSERT INTO "WORK_ON" VALUES (987987987, 30, 5.0)	1 row(s) inserted.	1
36	0.00	INSERT INTO "WORK_ON" VALUES (987654321, 30, 20.0)	1 row(s) inserted.	1
37	0.01	INSERT INTO "WORK_ON" VALUES (987654321, 20, 15.0)	1 row(s) inserted.	1
38	0.00	INSERT INTO "WORK_ON" VALUES (888665555, 20, NULL)	1 row(s) inserted.	1
39	0.02	INSERT INTO PROJECT VALUES (1, 'ProductX', 'Bellaire', 5)	1 row(s) inserted.	1
40	0.00	INSERT INTO PROJECT VALUES (2, 'ProductY', 'Sugarland', 5)	1 row(s) inserted.	1
41	0.00	INSERT INTO PROJECT VALUES (3, 'ProductZ', 'Houston', 5)	1 row(s) inserted.	1
42	0.01	INSERT INTO PROJECT VALUES (10, 'Computerization', 'Sta ord	1 row(s) inserted.	1
43	0.00	INSERT INTO PROJECT VALUES (20, 'Reorganization', 'Houston',	1 row(s) inserted.	1
44	0.00	INSERT INTO PROJECT VALUES (30, 'Newbene ts', 'Sta ord', 4	1 row(s) inserted.	1
45	0.02	INSERT INTO DEPENDENT VALUES ('Alice', 333445555, 'F', '04-0	1 row(s) inserted.	1
46	0.00	INSERT INTO DEPENDENT VALUES ('Theodore', 333445555, 'M', '1	1 row(s) inserted.	1
47	0.01	INSERT INTO DEPENDENT VALUES ('Joy', 333445555, 'F', '05-03-	1 row(s) inserted.	1
48	0.00	INSERT INTO DEPENDENT VALUES ('Abner', 987654321, 'M', '02-2	1 row(s) inserted.	1
49	0.00	INSERT INTO DEPENDENT VALUES ('Michael', 123456789, 'M', '01	1 row(s) inserted.	1
50	0.01	INSERT INTO DEPENDENT VALUES ('Alice', 123456789, 'F', '12-3	1 row(s) inserted.	1
51	0.00	INSERT INTO DEPENDENT VALUES ('Elizabeth', 123456789, 'F', '	1 row(s) inserted.	1
52	0.05	alter table employee add constraint employee_mgr foreign	Table altered.	0
53	0.02	alter table employee add constraint employee_dno foreign	Table altered.	0
54	0.02	alter table department add constraint department_mgr for	Table altered.	0
55	0.02	alter table dept_locations add constraint dept_locations	Table altered.	0
56	0.02	alter table project add constraint project_dnum foreign	Table altered.	0
57	0.02	alter table project add constraint project_plocation for	Table altered.	0
58	0.02	alter table work_on add constraint work_on_ssn foreign k	Table altered.	0
59	0.02	alter table work_on add constraint work_on_p_no foreign	Table altered.	0
60	0.01	alter table dependent add constraint dependent_ssn forei	Table altered.	0

row(s) 1 - 60 of 60

60	60	0
Statements Processed	Successful	With Errors

Figure 8: table2

6.2 Use basic select statements to provide rows of each table and provide screenshot.

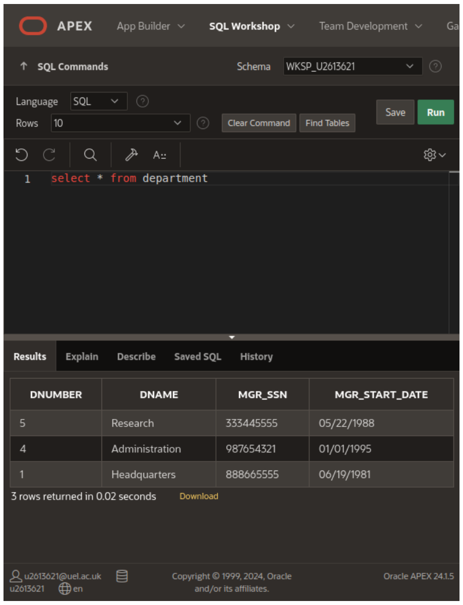


```
select * from employee
```

SSN	F_NAME	M_INIT	L_NAME	B_DATE	ADDRESS	SEX	SALARY	MGR_SSN	D_NO
123456789	John	B	Smith	01/09/1965	731 Fondren, Houston, TX	M	30000	333445555	5
333445555	Franklin	T	Wong	12/08/1955	638 Voss, Houston, TX	M	40000	888665555	5
999887777	Alicia	J	Zelaya	01/19/1968	3321 Castle, Spring, TX	F	25000	987654321	4
987654321	Jennifer	S	Wallace	06/20/1941	291 Berry, Bellaire, TX	F	43000	888665555	4
666884444	Ramesh	K	Narayan	09/15/1962	975 Fire Oak, Humble, TX	M	38000	333445555	5
453453453	Joyce	A	English	07/31/1972	5631 Rice, Houston, TX	F	25000	333445555	5
987987987	Ahmad	V	Jabbar	03/29/1969	980 Dallas, Houston, TX	M	25000	987654321	4
888665555	James	E	Borg	11/10/1937	450 Stone, Houston, TX	M	55000	-	1

8 rows returned in 0.05 seconds

Figure 9: table3

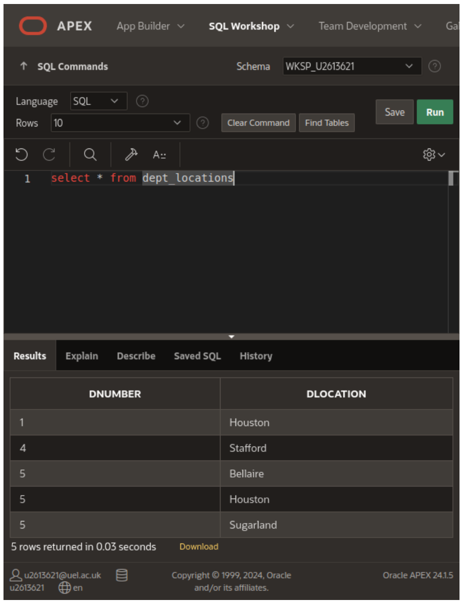


```
select * from department
```

DNUMBER	DNAME	MGR_SSN	MGR_START_DATE
5	Research	333445555	05/22/1988
4	Administration	987654321	01/01/1995
1	Headquarters	888665555	06/19/1981

3 rows returned in 0.02 seconds

Figure 10: Table 5

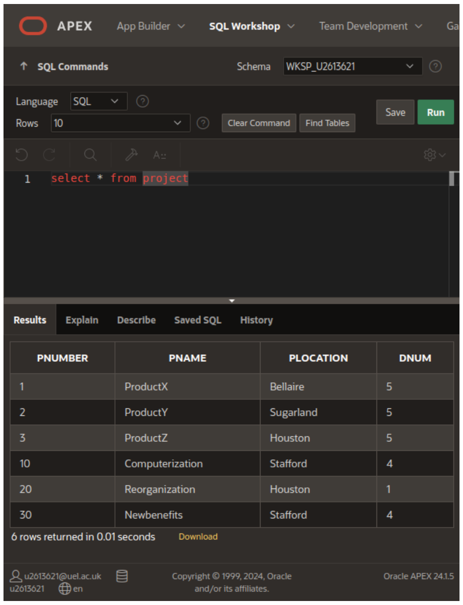


```
select * from dept_locations
```

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire
5	Houston
5	Sugarland

5 rows returned in 0.03 seconds

Figure 11: Table 6



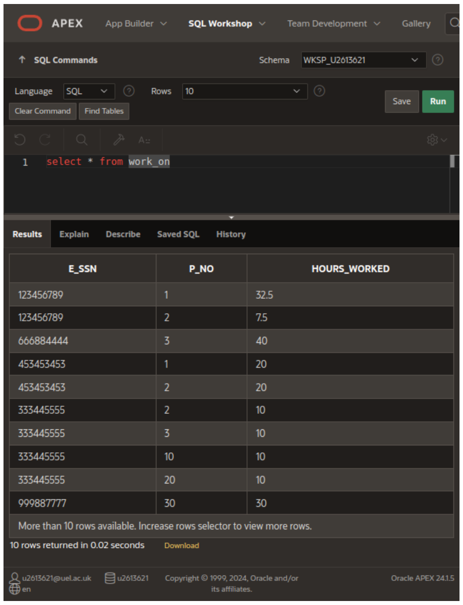
Schema: WKSP_U2613621
Language: SQL
Rows: 10

```sql
select * from project
```

PNUMBER	PNAME	PLOCATION	DNUM
1	ProductX	Bellaire	5
2	ProductY	Sugarland	5
3	ProductZ	Houston	5
10	Computerization	Stafford	4
20	Reorganization	Houston	1
30	Newbenefits	Stafford	4

6 rows returned in 0.01 seconds

Figure 12: Table 7



Schema: WKSP_U2613621
Language: SQL
Rows: 10

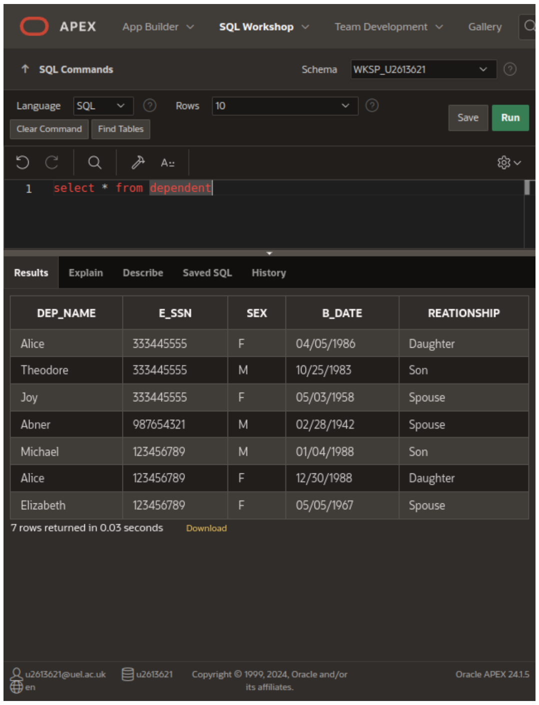
```sql
select * from work_on
```

E_SSN	P_NO	HOURS_WORKED
123456789	1	32.5
123456789	2	7.5
666884444	3	40
453453453	1	20
453453453	2	20
333445555	2	10
333445555	3	10
333445555	10	10
333445555	20	10
999887777	30	30

More than 10 rows available. Increase rows selector to view more rows.

10 rows returned in 0.02 seconds

Figure 13: Table 8



Schema: WKSP_U2613621
Language: SQL
Rows: 10

```sql
select * from dependent
```

DEP_NAME	E_SSN	SEX	B_DATE	REATIONSHIP
Alice	333445555	F	04/05/1986	Daughter
Theodore	333445555	M	10/25/1983	Son
Joy	333445555	F	05/03/1958	Spouse
Abner	987654321	M	02/28/1942	Spouse
Michael	123456789	M	01/04/1988	Son
Alice	123456789	F	12/30/1988	Daughter
Elizabeth	123456789	F	05/05/1967	Spouse

7 rows returned in 0.03 seconds

Figure 14: table9

7 Week 7

7.1 Queries

7.1.1 Create a SQL statement that displays only the first_name and salary of an employee whose salary is between 30,000 and 40000.

```sql
SELECT fname, salary
FROM employee
WHERE salary >= 30000 AND salary <= 40000;
```

7.1.2 Create a SQL statement that displays the first_name and last_name of employees whose last_name is either ‘Smith’, ‘King’, or ‘Rogers’.

```sql
select fname, lname
from employee
where lname in ('Smith','King','Rogers');
```

7.1.3 Create a SQL statement that displays the first_name and last_name of employees whose last_name starts with ’S’.

```sql
select fname, lname
from employee
where lname like 'S%';
```

7.1.4 For each department, retrieve the department number, the number of employees in the department, and their average salary.

```sql
select dno,
count(ssn) as no_of_employees,
avg(salary) as average_salary
from employee
group by dno
order by dno asc;
```

7.1.5 For each project, retrieve the project number, the project name, and the number of employees who work on that project.

```
select works_on.pno, count(works_on.essn) as employees_on_project, project.pname
from works_on
join project on works_on.pno = project.pnumber
GROUP BY works_on.pno, project.pname
Order by works_on.pno asc;
```

7.1.6 For each project on which more than two employees work, retrieve the project number, the project name, and the number of employees who work on the project.

```
select works_on.pno, count(works_on.essn) as employees_on_project, project.pname
from works_on
join project on works_on.pno = project.pnumber
GROUP BY works_on.pno, project.pname
having count(works_on.essn) > 2
Order by works_on.pno asc;
```

7.1.7 Retrieve the name and address of all employees who work for the 'Research' department without using join

```
select fname ||' '|| lname as Employee_Name, address
from employee
where dno in (
    select dnumber
    from department
    where dname = 'Research'
);
```

7.1.8 Modify query 1 and observe results by using join

```
select fname ||' '|| lname as Employee_Name, address
from employee
join department on department.dnumber = employee.dno
where dname = 'Research'
```

7.1.9 Modify query 1 and use subquery instead of join

```
select fname ||' '|| lname as Employee_Name, address
from employee
where dno in (
    select dnumber
    from department
    where dname = 'Research'
);
```

7.1.10 Retrieve the last name of employees and their supervisors (using inner join)

```
select employee.lname as Employee_Last_Name, mgr.lname as Manager_Last_Name
from employee
inner join employee mgr
    on employee.superssn = mgr.ssn
order by Employee_Last_Name asc;
```

7.1.11 Modify query 4 to display all employees with supervisor and also that employee where a supervisor is not assigned (left outer join) Note the difference in results by increasing row counts in the output

```
SELECT e.lname AS Employee_Last_name, s.lname AS Supervisor_Last_Name
FROM employee e
LEFT OUTER JOIN
    employee s
    ON e.superssn = s.ssn
Order by Employee_Last_Name;
```

7.1.12 Modify query 4 to display all the supervisors with and without employees assigned to them (Right outer join)

```
SELECT e.fname AS Employee_Last_name, s.fname AS Supervisor_Last_Name
FROM employee e
right outer JOIN
    employee s
    ON e.superssn = s.ssn
Order by Employee_Last_Name;
```

7.1.13 Modify query 4 to display all employees with and without supervisors and all supervisors with and without employees (Full outer join)

```
SELECT e.fname AS Employee_Last_name, s.fname AS Supervisor_Last_Name
FROM employee e
full outer JOIN
    employee s
    ON e.superssn = s.ssn
Order by Employee_Last_Name;
```

7.2 Creating Views

7.2.1 Create a view to display employee name and their salaries who work for the research department.

```
CREATE VIEW RESEARCH_EMPLOYEE_SALARIES AS
SELECT FNAME ||' '|| LNAME as Employee_Name , SALARY
FROM EMPLOYEE
JOIN DEPARTMENT ON EMPLOYEE.DNO = DEPARTMENT.DNUMBER
WHERE DEPARTMENT.DNAME = 'Research';

select *
from RESEARCH_EMPLOYEE_SALARIES;
```

7.2.2 Create a view to display the employee name and the name of the project and project hour in which each employee works. Provide a screenshot of the code and results

```
CREATE VIEW RESEARCH_EMPLOYEE_PROJECTS AS
SELECT FNAME ||' '|| LNAME as Employee_Name , PROJECT.PNAME, WORKS_ON.HOURS
FROM EMPLOYEE
JOIN WORKS_ON ON EMPLOYEE.SSN = WORKS_ON.ESSN
JOIN PROJECT ON WORKS_ON.PNO = PROJECT.PNUMBER
```

```
ORDER BY PROJECT.PNUMBER;

select *
from RESEARCH_EMPLOYEE_PROJECTS;
```

7.3 Creating and Executing Procedures

7.3.1 Download 1a - Raise SalaryProcedure, and run in APEX. Understand the procedure logic

```
CREATE OR REPLACE PROCEDURE raise_salary
(
    employee_ssn IN CHAR,
    employee_pct IN NUMBER DEFAULT 5,
    result_message OUT CHAR
)
AS
    old_salary employee.salary%TYPE;
    increase_amount NUMBER;

/*
Program-defined exceptions are declared here and are used to identify
exception events which will interrupt main program execution.
*/
    pct_too_high EXCEPTION;
    update_error EXCEPTION;

BEGIN
    --Disallow raises which exceed 50% on the basis of the business rules.
    IF employee_pct > 50 THEN
        RAISE pct_too_high;
    END IF;

    --Retrieve the salary from the employee table
    SELECT salary
    INTO old_salary
    FROM employee
    WHERE ssn = employee_ssn;

/*
```

```
If the existing salary is unknown or NULL, or if it is 0, then no raise is
possible. Otherwise, compute the raise amount and issue an update to the
database.
*/
    IF (old_salary IS NOT NULL) AND (old_salary > 0) THEN

        --Convert the employee pct parameter value to a
        --numeric percentage and update the appropriate
        --database column.
        increase_amount := employee_pct / 100;

        UPDATE employee
        SET salary = salary + (salary * increase_amount)
        WHERE ssn = employee_ssn;

        IF SQL%ROWCOUNT <> 1 THEN
            RAISE update_error;
        END IF;

        --Set the output parameter value if necessary.
    ELSE

        --Set the message
        result_message := 'Current salary is either NULL or 0';

    END IF;

EXCEPTION
/*
Set the output parameter value here as well based upon program-defined and
system-defined exceptions which might occur.
*/
    WHEN pct_too_high THEN
        result_message := 'Raise percentage may not exceed 50%';

    WHEN NO_DATA_FOUND THEN
        result_message := 'Employee ' || employee_ssn || ' not found';

    WHEN update_error THEN
        result_message := 'Database error';

    WHEN OTHERS THEN
```

```
        result_message := 'Unknown error';

END raise_salary;
```

7.3.2 Execute the procedure using 1b- Fire Raise Salary procedure, to test output. Provide different values in the test procedure to test each statement in the program logic. For each test statement provide a screenshot of the code and execution results

```
DECLARE

  --Declare variables
  output_text CHAR(100);

BEGIN

  --Fire raise_salary procedure
  raise_salary('123456789', 48, output_text);

  --Output the results
  dbms_output.put_line('Output Text: ' || output_text );

END;
```

7.4 Creating and Executing Functions

7.4.1 Download 2a - Salary Valid function, and run in APEX. Understand the function logic

```
CREATE OR REPLACE FUNCTION salary_valid
(
    input_ssn     IN CHAR,
    input_salary    IN NUMBER
)
RETURN boolean
IS
    count_management  NUMBER;
    count_projects    NUMBER;
    count_dependents    NUMBER;
    salary_limit        NUMBER;
BEGIN

    salary_limit := 50000;


    SELECT count(*)
    INTO count_management
    FROM department
    WHERE department.mgrssn = input_ssn;

  --Test the count_management value
    IF count_management > 0 THEN
        salary_limit := salary_limit + 1000;
    END IF;

    SELECT count(*)
    INTO count_projects
    FROM works_on
    WHERE works_on.essn = input_ssn;

  --Recalculate the salary limit
    salary_limit := salary_limit + (count_projects * 2000);

    SELECT count(*)
    INTO count_dependents
    FROM dependent
    WHERE dependent.essn = input_ssn;

  --Recalculate the salary limit
    salary_limit := salary_limit + (count_dependents * 3000);

    IF input_salary > salary_limit THEN
        RETURN (FALSE);
    ELSE
        RETURN (TRUE);
    END IF;

END salary_valid;
```

7.4.2 Execute the function 2b - Test Salary_Valid Function to test output. Provide different values in the test procedure to test each statement in the program logic. For each test statement provide a shot of the code and execution results.

```
SET SERVEROUTPUT ON;

BEGIN

  --Test the Salary_Valid function
  IF salary_valid('123456789', 80000) THEN
    dbms_output.put_line('Salary is valid');
  ELSE
    dbms_output.put_line('Salary is not valid');
  END IF;

END;
```

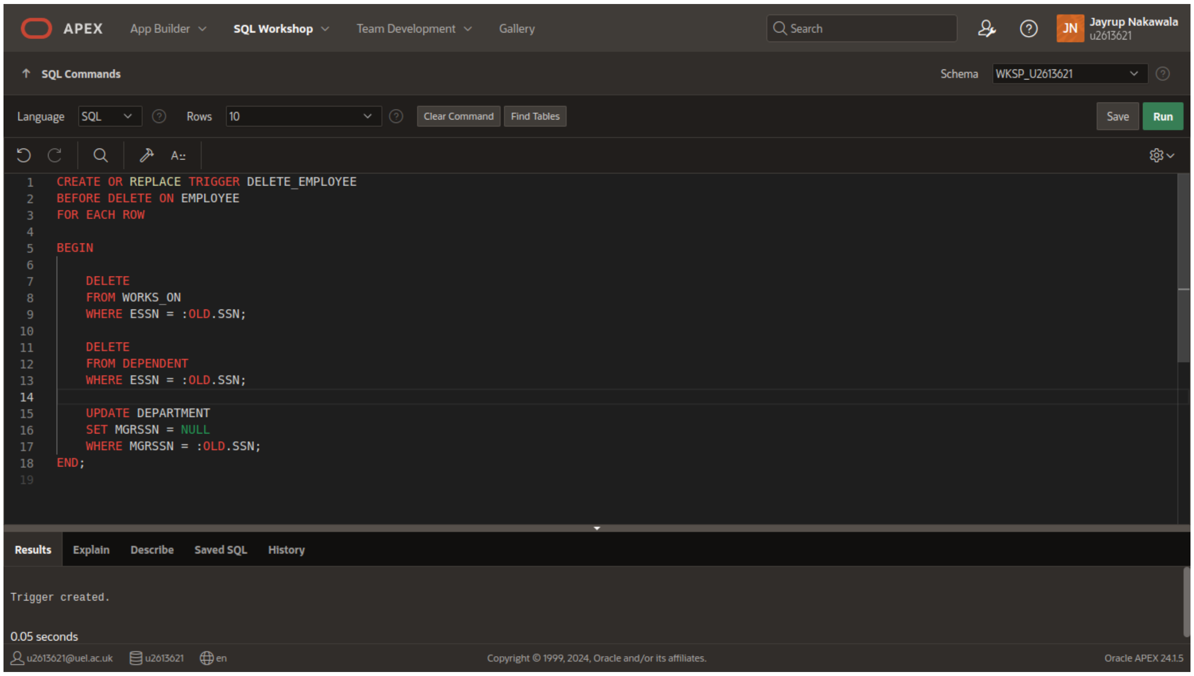
8 Week 8

8.1 Consider the COMPANY database provided in the previous assignments. Using the syntax of Oracle triggers, create triggers to do the following:

Whenever an employee is deleted, delete the PROJECT tuples and DEPENDENT tuples related to that employee, and if the employee manages a department or supervises employees, set the Mgr_ssn for that department to NULL and set the Super_ssn for those employees to NULL.

You are required to provide the trigger code, SQL statements to test the triggers and screenshots of the database state of affected rows before and after executing trigger test statements and an explanation of your logic.

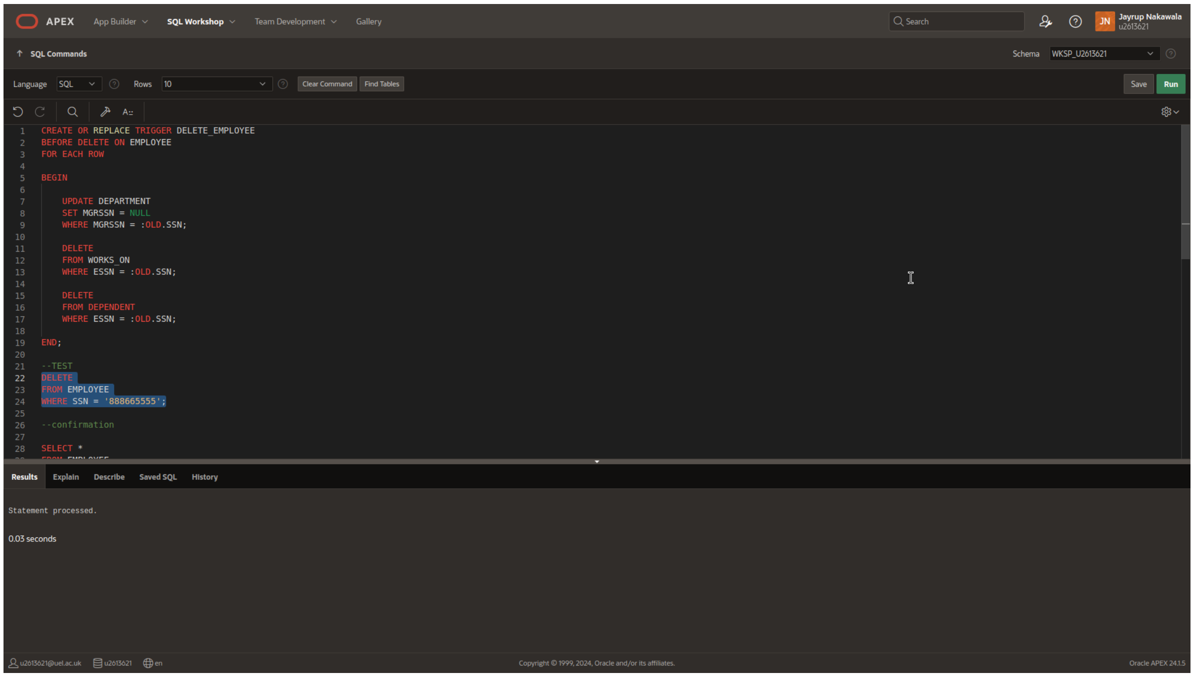
```
CREATE OR REPLACE TRIGGER DELETE_EMPLOYEE
BEFORE DELETE ON EMPLOYEE
FOR EACH ROW

BEGIN

    UPDATE DEPARTMENT
    SET MGRSSN = NULL
    WHERE MGRSSN = :OLD.SSN;
    DELETE
    FROM WORKS_ON
    WHERE ESSN = :OLD.SSN;

    DELETE
    FROM DEPENDENT
    WHERE ESSN = :OLD.SSN;

END;

--TEST
DELETE
FROM EMPLOYEE
WHERE SSN = '888665555';

--confirmation

SELECT *
FROM EMPLOYEE
WHERE SSN = '888665555';

SELECT *
FROM dependent
WHERE eSSN = '888665555';

SELECT *
FROM works_on
WHERE eSSN = '888665555';

SELECT *
FROM department
WHERE MGRSSN = '888665555';
```



```sql
CREATE OR REPLACE TRIGGER DELETE_EMPLOYEE
BEFORE DELETE ON EMPLOYEE
FOR EACH ROW

BEGIN

    DELETE
    FROM WORKS_ON
    WHERE ESSN = :OLD.SSN;

    DELETE
    FROM DEPENDENT
    WHERE ESSN = :OLD.SSN;

    UPDATE DEPARTMENT
    SET MGRSSN = NULL
    WHERE MGRSSN = :OLD.SSN;
END;
```

```
Trigger created.

0.05 seconds
```

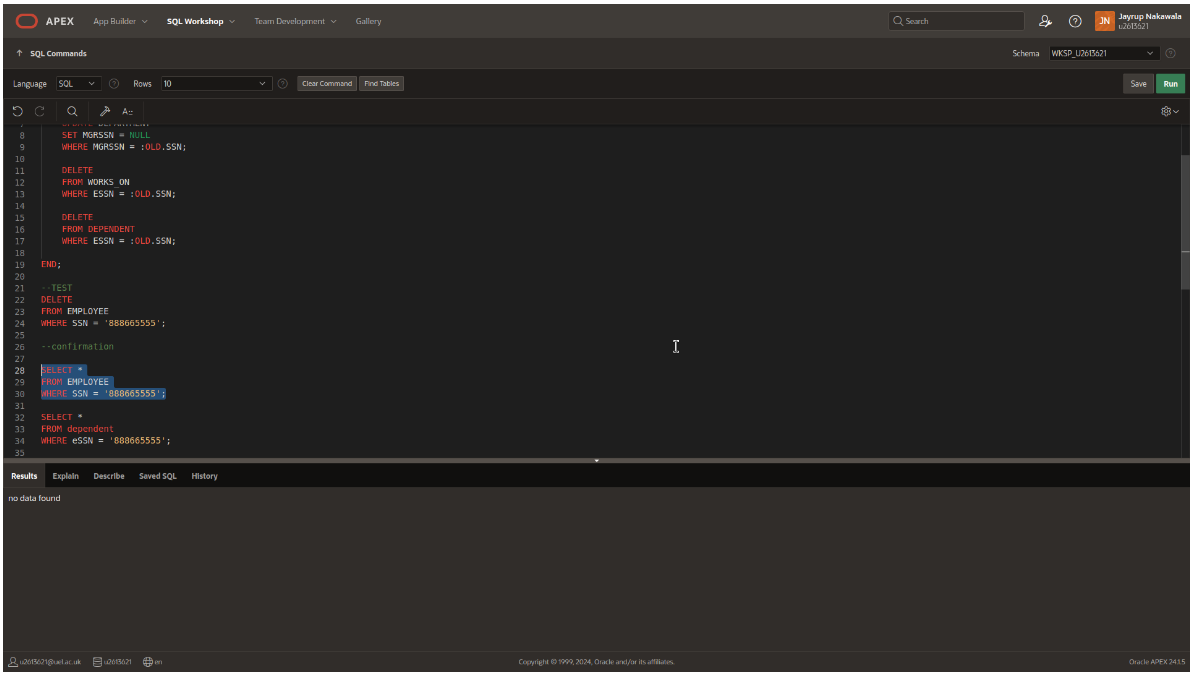
Figure 15: Trigger creation



```sql
CREATE OR REPLACE TRIGGER DELETE_EMPLOYEE
BEFORE DELETE ON EMPLOYEE
FOR EACH ROW

BEGIN

    UPDATE DEPARTMENT
    SET MGRSSN = NULL
    WHERE MGRSSN = :OLD.SSN;

    DELETE
    FROM WORKS_ON
    WHERE ESSN = :OLD.SSN;

    DELETE
    FROM DEPENDENT
    WHERE ESSN = :OLD.SSN;

END;

--TEST
DELETE
FROM EMPLOYEE
WHERE SSN = '888665555';

--confirmation

SELECT *
```

```
Statement processed.

0.03 seconds
```

Figure 16: Testing the Trigger



```
        SET MGRSSN = NULL
        WHERE MGRSSN = :OLD.SSN;

        DELETE
        FROM WORKS_ON
        WHERE ESSN = :OLD.SSN;

        DELETE
        FROM DEPENDENT
        WHERE ESSN = :OLD.SSN;

    END;

    --TEST
    DELETE
    FROM EMPLOYEE
    WHERE SSN = '888665555';

    --confirmation

    SELECT *
    FROM EMPLOYEE
    WHERE SSN = '888665555';

    SELECT *
    FROM dependent
    WHERE eSSN = '888665555';
```

```
no data found
```

Figure 17: Confirming that the trigger works

9 Week 9

9.1 What is a Transaction in the context of Databases?

A Transaction is a logical unit of work performed on the database. It takes the database from one consistent state to another consistent state, i.e., duplicate data does not conflict and the Database follows the rules defined.

9.2 What is ACID in Transaction? Define each identified.

ACID stands for:

- **Atomicity**: a transaction is either performed whole or it is not performed at all.
- **Consistency preservation**: a transaction takes the database from one consistent state to another.
- **Isolation**: a transaction is executed independently of other concurrent transactions, ensuring they do not interfere.

- **Durability or permanency**: once a transaction is committed, these changes must never be lost due to subsequent failure.

9.3 What are the different states of a transaction?

These are the following states of a transaction:

- **Active**: this is when the transaction is actively being performed. Read/write occurs at this state.
- **Partially committed**: After the transaction has ended, it enters this state where the transaction still be aborted.
- **Commited**: this state is achived after successfully committing the transaction.
- **Failed**: When a transaction is aborted, either in the active state or the partially committed state, it enters the failed state.
- **Terminated**: This is the final state where the transaction ends. Which can be achived from the failed state or the committed state.

9.4 Define COMMIT and ROLLBACK and give example of each.

- **Commit**: This signals that the transaction has been completed successfully so that all the changes/update done to the data item should be saved and safely committed to the database.
- **Rollback**: This signals that the transaction has not been successful and so that any changes/updates done by the transaction should be undone and not saved into the database.

9.5 What are the three main recovery techniques?

- **Immediate Update**: In this technique, the data item is update in the backup as soon as it is modified in the cache.
- **Deferred Update**: In this technique, the modified data item in the cache is updated in the backup after a certain transaction ends or after a certain number of transactions have been completed.
- **Shadow Update**: In this technique, the modified version of a data item does not override the disk copy instead, it is stored in a separate disk.

10 Week 10

10.1 What is the purpose of database security in general?

The main purpose of database security is protecting the database from unauthorized access. The database may contain sensitive inforamtion, whether related to personal details or users or projects with stratetgic importance.

10.2 Describe the approaches for securing DBMSs on the Web.

There are multiple approaches that can be taken to secure DBMSs on the web, some of them are:

- **Privacy**: when transmitting information, it must not be accessible by anoyone other than the sender and the reciever.
- **Integrity**: the information should not be changged during transmission
- **Authenticity**: The reciever should be sure that the infotmation came from the sender.

To make sure these practices are ensured, we take help of technologies like proxy servers, firewalls, encryption, SSL certificates, etc.

10.3 Various threats exist on computer system, describe the possible threats and the likelihood of occurring for the followings:

- **Hardware Threats**: Physical damage (fire, water), theft, hardware failure, power issues.
  - Likelihood: Moderate for damage/failure, high for theft in unsecured areas, and moderate for power issues.
- **DBMS and Application Software Threats**: Software bugs, malware, unauthorized access, outdated versions.
  - Likelihood: High for bugs, moderate for malware and access issues, low with updates.
- **Database Threats**: Data breaches, corruption, loss, SQL injection.
  - Likelihood: High for breaches, moderate for loss and corruption, depends on security measures.

- **Communication Networks Threats**: Eavesdropping, DoS, MitM attacks, data leakage.
  - Likelihood: High without encryption, moderate for DoS/MitM, mitigated with security protocols. Users of the System
- **End Users**: Phishing, unintentional errors.
- **Programmers/Operators**: Malicious actions, coding flaws.
- **Admins**: Misuse of privileges, configuration errors.
  - Likelihood: High for user errors, moderate for insider threats.

10.4 What is the purpose of access control matrix? Illustrate your answer with an example.

The Access Control Matrix is used to define and manage the permissions of various users (subjects) to access specific database objects (e.g., tables, files). It maps users to the actions they are allowed to perform (e.g., read, write, update). Example:

Subject/User	Table 1	Table 2
User 1	Read	Read
User 2	Update	Read
User 3	Read, Write	Update

In this example, the matrix defines which operations users can perform on tables, ensuring unauthorized access is prevented.

10.5 Explain how views can be used to improve security of DBMS.

A view is a virtual table that only provides access to the subset of datatbase which the admin allows. Views hide sensitive columns or rows by limiting what the user can see or manipulate, thereby enhancing security.

10.6 What is difference between Discretionary Access Control (DAC) and Mandatory Access Control (MAC)

	Discretionary Access Control (DAC)	Mandatory Access Control (MAC)
Defination	Users have control over their objects and can decide who can access them.	System-enforced access based on predefined policies, not user discretion.
Flexibility:	High; decisions are at the user's discretion.	Low; policies are rigid and centralized.
Risk:	Susceptible to insider threats due to user control.	More secure, with less reliance on user decisions.
Example:	A user grants or revokes privileges on a table using SQL commands like GRANT and REVOKE.	Classified systems where users can only access data they are cleared for, based on security labels.