CIS 2268: Final Project

Tyler Craig
SINCLAIR COMMUNITY COLLEGE | FALL 2020

SINCLAIR
COMMUNITY COLLEGE

Contents

CIS 2268: Final Project

SINCLAIR COMMUNITY COLLEGE | FALL 2020

DATABASE DESCRIPTION

In production environments down-time needs to be limited as much as possible. To help with this, many organizations implement a type of maintenance tracking database. Usually, users can create new maintenance tasks, assign workers to the tasks, and then provide details such as cost or total hours worked performing the assigned task. These databases allow workers to view maintenance tasks and plan production events around any failed piece of equipment. Typically, these databases also allow preventative maintenance tasks to be tracked. The task can be configured as a re-occurring event, and notifications are sent several days before the task is due.

The database created for the midterm aims to mimic the basics of a maintenance database. The database allows users to create new maintenance tasks, assign workers to a task, and assign equipment pieces to the maintenance tasks. Using various tables, when maintenance tasks are assigned, users will be able to view the task ID, the employee assigned to the task, the employee's position, the piece of equipment, and various information about the piece of equipment (serial number, product number, manufacturer). Connecting equipment pieces to manufacturers will help employees order replacement parts if needed or consult reference manuals. Since the TASKS table is the only table that will hold 'dynamic' data, this is the only table that regular users will be able to interact with. When a task is created, regular users will be able to add information into any of the fields in the TASKS table. However, to ensure the database is accurate, only administrators will be able to UPDATE information in the fields (only an administrator could change fields like cost/totalhrs to a different value).

Ideally, users would interact with the database through a web-interface. The web-interface would have more intuitive controls than using SQL commands. The controls on the web-interface would be mapped to stored procedures on the database back-end, and corresponding results would be returned and displayed on the web-interface. Ideally, this database would target maintenance crews such as technicians, engineers, operators. Of course, queries could be performed by users that would shows statistics such as 'total maintenance time per equipment', 'total maintenance cost per equipment', or 'maintenance category performed most per equipment'. These queries would give valuable insight into how equipment is performing, and where money should be spent for replacements.

LIMITATIONS

The database provides a rudimentary method of tracking maintenance tasks. There are several flaws in the database:

- Manufacturers address needs to be more accurate, provide zip-code and if in US provide a state
- For locations, could provide a separate table for facility addresses, instead of naming the facility provide a facility code. This would link to a facility_code table that would provide more details about the facility
- Status on equipment could have a status code, instead of 'Active' or 'Down'. Could provide more detail and help determine different stages of maintenance equipment is in. Statuses could be like 'Running', 'Running, but needs maintenance', 'Preventative Maintenance scheduled', etc.
- Only one person per maintenance task, usually several employees are involved in one maintenance task, currently no way to show multiple employees per task
- Different character limitations on several fields
- Should use a 'Problem' field in the TASKS table where createdBy employee could put description of maintenance or issue. The comments section could then be used by assignedTo employee to describe solution
- Could put 'Technical Document' or 'Reference Manual' field into the EQUIPMENT table. Manuals could be tied to individual pieces of equipment, so user isn't constantly searching internet for different manuals

DML / DDL / TCS STATEMENTS USED

Below show some examples of the DDL, DML, and TCS statements used throughout the project. For conciseness not all statements will be listed.

TASKS

DDL

- CREATE TABLE tasks
- DROP TABLE tasks

DML

- INSERT INTO tasks VALUES (taskID, equipmentID, …)
- UPDATE tasks SET (values = values) WHERE field = value
- DELETE tasks WHERE field = value

TCS

- ROLLBACK
- COMMIT

MANUFACTURERS

DDL

- CREATE TABLE manufacturers
- DROP TABLE manufacturers

DML

- INSERT INTO manufacturers VALUES (mfg_code, manufacturer, …)
- UPDATE manufacturers SET (values = values) WHERE field = value
- DELETE manufacturers WHERE field = value

EQUIPMENT

DDL

- CREATE TABLE equipment
- DROP TABLE equipment

DML

- INSERT INTO equipment VALUES (equipmentID, description, status…)
- UPDATE equipment SET (values = values) WHERE field = value
- DELETE equipment WHERE field = value

TCS

- COMMIT
- ROLLBACK

TASKS SUPPORTED (BY TABLE)

TASKS

The TASKS table will be the main table users will interact with. It will hold the information such as employee assignment, taskID, task cost, task category, etc. Users will use the TASKS table to view maintenance tasks assigned to them, or to make new maintenance tasks. Maintenance costs and total time used for maintenance will also be tracked by each maintenance task. Since each task will have a total cost/total time field associated with it, users can generate reports on tasks that are taking the longest time, or tasks that cost the most to perform. Since each task is tied to a piece of equipment, this could even be broken type by equipment.

Regular users will be able to add any information into all the fields of the table. Once the task is created, only ADMINISTRATORS will be able to modify the cost, totalhrs, createdBy, and assignedTo fields.

EQUIPMENT

The EQUIPMENT table will hold all equipment information. Once a maintenance task is assigned, the task is assigned to a piece of equipment in the EQUIPMENT table. Users can reference the equipment table to view information such as serial number, product number, manufacturer code, purchase price, etc. Using the equipment ID from the TASKS table and the equipment ID from the

EQUIPMENT table, users know what equipment needs to be serviced and what equipment is currently working.

Regular will NOT have the ability to modify this table. This table will be used mainly for reference, and all details should be kept the same as when they were first entered. If changes need to be made to the table, only administrators will be able to modify the table. This will help ensure the integrity of the data.

MANUFACTURERS

Like the EQUIPMENT table, the MANUFACTURERS table will mainly be used for reference. When a piece of equipment is created, it will be assigned a manufacturer code. If the manufacturer is not in the MANUFACTURERS table, a new manufacturer code will need to be generated. The user will be able to view the name, address, and website of the manufacturer. Using this information could be helpful when users need to find technical documents or order replacement parts.

Because this data should not change after creation, regular users will not be able to modify this table. Only administrators will be able to make changes to the table. Administrators will be allowed to modify any of the field information, and they will be able to delete manufacturers if needed (likely company policy would require reason for mfg deletion, like mfg went out of business or all equipment from mfg was sold).

TASK CATEGORY

The TASK_CATEGORY table will be used to help categorize maintenance tasks. When a maintenance task is created, it should be assigned a category code from the employee that created the task. The category code will be used to reference a description of the category. This will help the user what type of work should be expected/should be completed. Implementing a category code into the TASKS table, users will have insight into what tasks are being performed, and on what pieces of equipment. This could help the company replace aging equipment or determine where possible problems might arise in the future (if a maintenance task was created with a ‘CONFIGURATION-CHANGE, users could start troubleshooting with older configurations).

Regular users will not be able to modify this table, seeing as it should only be used for reference. To ensure data integrity across the TASK and TASK_CATEGORY tables, administrators are the only group that will allowed to make changes to the table.

EMPLOYEES

When a maintenance task is created, an employee IDs are used. An employee ID is used in the createdBy field of the task. The createdBy field is used to determine who created the task. Any questions about why the task was created should be directed to the employee in the createdBy field. The assignedTo field also requires an employee ID. The assignedTo field is used to assign tasks to employees. This is the employee that is responsible for completing the task.

The EMPLOYEES table will hold information about different employees. The employees name, phone number, email address, start date, and a position code can be used. This information will be sufficient in determining who assigned a task, and the most competent person to complete a task.

Only administrators will be able to make modifications to this table.

POSITION_CODES

Each employee in the EMPLOYEES table will be assigned a position_code. The position_code will be used to reference the POSITION_CODES table, where position_codes align with an actual description title. Essentially, employees might have a position code of 'EE001', when referencing the POSITION_CODES table, this would align with the position 'ElectricalEngineer'.

Only administrators will be able to make modifications to this table.

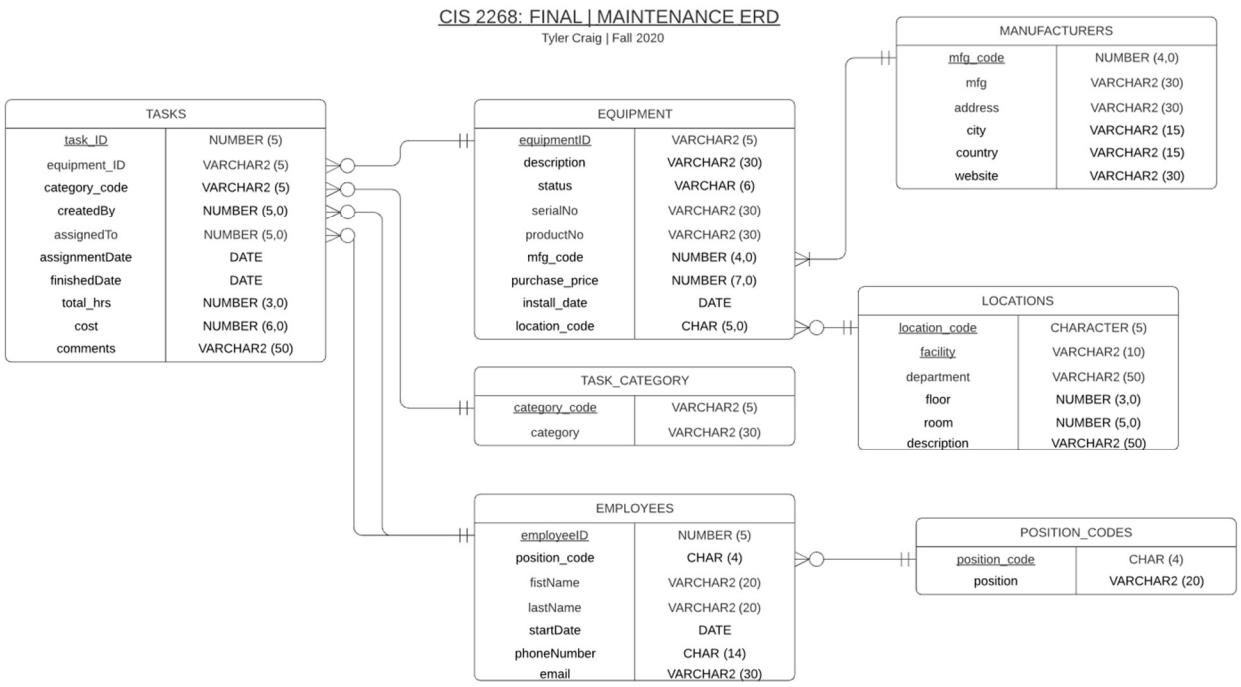
LOCATIONS

When a record equipment is created, the equipment record must have a location_code. The location_code will be used to determine the physical location of the equipment. When assigning maintenance tasks, users can view the equipment that is attached to the task, and view the location_code. The location_code will reference a location inside the LOCATIONS table. Information such as the facility, room number, floor number, and department can be viewed using the location_code.

Only administrators will be able to make modifications to this table.

Screenshots

ITEM 1: At Least 7 Tables All Linked



```
/*################################################CREATE MANUFACTURERS TABLE#############################################################*/
CREATE TABLE manufacturers
(   mfg_code NUMBER (4),
    mfg VARCHAR2 (30) NOT NULL,
    address VARCHAR2 (30),
    city VARCHAR2 (15),
    country VARCHAR2 (15),
    website VARCHAR2 (30) NOT NULL,

        CONSTRAINT manufacturers_mfgCode_pk PRIMARY KEY (mfg_code),
        CONSTRAINT manufacturers_mfg_uq UNIQUE (mfg));
```

```
/*################################################CREATE LOCATIONS TABLE#################################################################*/
CREATE TABLE locations
(   location_code CHARACTER (5),
    facility VARCHAR2 (10) NOT NULL,
    department VARCHAR2 (50),
    floor NUMBER (3) NOT NULL,
    room NUMBER (5) NOT NULL,
    description VARCHAR2 (50),

        CONSTRAINT locations_CodeFacility_pk PRIMARY KEY (location_code, facility));

INSERT INTO locations
    VALUES ('EN000', 'Site A', 'Engineering', 000, 00002, 'Eng Lab 2');

INSERT INTO locations
    VALUES ('AI005', 'Site A', 'AI Development', 005, 00510, 'AI Dev Lab 10');

INSERT INTO locations
    VALUES ('PT001', 'Site C', 'Propulsion Testing', 001, 00102, 'Prop Testing Area 2');

INSERT INTO locations
    VALUES ('CC001', 'Site A', 'Central Control', 001, 00101, 'Site A Central Control');

INSERT INTO locations (location_code, facility, floor, room)
    VALUES ('MB001', 'Site A', '000', '00001');
```

```
/*###############################################CREATE POSITION_CODES TABLE#############################################################*/
CREATE TABLE position_codes
(   position_code CHAR (4),
    position VARCHAR2 (20) NOT NULL,

        CONSTRAINT positionCodes_positionCode_pk PRIMARY KEY (position_code),
        CONSTRAINT positionCodes_position_uq UNIQUE (position));
```

```
/*################################################CREATE EMPLOYEES TABLE#################################################################*/
CREATE TABLE employees
(   employeeID NUMBER (5),
    position_code CHAR (4),
    firstName VARCHAR2 (20) NOT NULL,
    lastName VARCHAR2 (20) NOT NULL,
    startDate DATE DEFAULT SYSDATE,
    phoneNumber CHARACTER (14),
    email VARCHAR2 (30) NOT NULL,

        CONSTRAINT employees_employeeID_pk PRIMARY KEY (employeeID),
        CONSTRAINT employees_positionCode_fk FOREIGN KEY (position_code)
            REFERENCES position_codes (position_code),
        CONSTRAINT employees_phoneNumber_uq UNIQUE (phoneNumber),
        CONSTRAINT employees_email_uq UNIQUE (email));
```

```
/*################################################CREATE EQUIPMENT TABLE#############################################################################*/
CREATE TABLE equipment
(   equipmentID VARCHAR2 (5),
    description VARCHAR2 (30),
    status VARCHAR2 (6) DEFAULT 'Active',
    serialNo VARCHAR2 (30) NOT NULL,
    productNo VARCHAR2 (30) NOT NULL,
    mfg_code NUMBER (4),
    purchase_price NUMBER (7),
    install_date DATE DEFAULT SYSDATE,
    location_code CHARACTER (5) NOT NULL,

        CONSTRAINT equipment_equipmentID_pk PRIMARY KEY (equipmentID),
        CONSTRAINT equipment_status_ck CHECK (status IN ('Active', 'Down')),
        CONSTRAINT equipment_serialNo_uq UNIQUE (serialNo),
        CONSTRAINT equipment_mfgCode_fk FOREIGN KEY (mfg_code)
            REFERENCES manufacturers (mfg_code));
```

```
/*################################################CREATE TASK_CATEGORY TABLE#########################################################################*/
CREATE TABLE task_category
(   category_code VARCHAR2 (5),
    category VARCHAR2 (30) NOT NULL,

        CONSTRAINT taskCategory_categoryCode_pk PRIMARY KEY (category_code),
        CONSTRAINT taskCategory_category_uq UNIQUE (category));
```

```
/*################################################CREATE TASKS TABLE#################################################################################*/
CREATE TABLE tasks
(   taskID NUMBER (5),
    equipmentID VARCHAR2 (5),
    category_code VARCHAR2 (5),
    createdBy NUMBER (5),
    assignedTo NUMBER (5),
    assignmentDate DATE DEFAULT SYSDATE NOT NULL,
    finishedDate DATE,
    totalHrs NUMBER (3),
    cost NUMBER (6),
    comments VARCHAR2 (50),

        CONSTRAINT tasks_taskID_pk PRIMARY KEY (taskID),
        CONSTRAINT tasks_categoryCode_fk FOREIGN KEY (category_code)
            REFERENCES task_category (category_code),
        CONSTRAINT tasks_equipmentID_fk FOREIGN KEY (equipmentID)
            REFERENCES equipment (equipmentID),
        CONSTRAINT tasks_createdBy_fk FOREIGN KEY (createdBy)
            REFERENCES employees (employeeID),
        CONSTRAINT tasks_assignedTo_fk FOREIGN KEY (createdBy)
            REFERENCES employees (employeeID));
```

ITEM 2: At Least 4 Tables Have 10 Records

EMPLOYEES TABLE

EMPLOYEEID	POSITION_CODE	FIRSTNAME	LASTNAME	STARTDATE	PHONENUMBER	EMAIL
12345	ME01	Amanda	Ripley	02-JAN-10	(123)456-7890	aripley@blueprupulsion.com
23456	EE01	Issac	Clarke	02-JAN-10	(234)561-2389	iclarke@blueprupulsion.com
992	CE01	Damon	Baird	03-FEB-12	(123)412-3145	dbaird@blueprupulsion.com
12	ME02	Carter	Matthews	04-FEB-12	(324)123-4567	cmatthews@blueprupulsion.com
13	SE01	Frank	Jacobs	08-APR-12	(222)111-3333	fjacobs@blueprupulsion.com
10699	CE01	Yuki	Hamara	08-APR-12	(456)123-6734	yhamara@blueprupulsion.com
10698	QI01	Samir	Nadir	09-MAY-12	(333)245-5647	snadir@blueprupulsion.com
70912	SD01	Hal	Kyles	05-JUN-12	(555)666-7878	hkyles@blueprupulsion.com
10234	SD02	Jimmy	Buffett	09-JUN-12	(555)555-5555	jbuffett@blueprupulsion.com
233	TC02	Wess	Brookes	10-JUN-12	(999)888-7777	wbrookes@blueprupulsion.com
232	ME01	Homer	Hickam	11-JUN-12	(332)555-6767	hhickam@blueprupulsion.com

MANUFACTURERS TABLE

MFG_CODE	MFG	ADDRESS	CITY	COUNTRY	WEBSITE
1000	Weyland-Yutani	1131 Kings Street	London	United Kingdom	weylandyutani.com
1001	Cyberdyne Systems	2341 Infinity Way	Houston	United States	skynet.net
1003	Wayne Industries	267 Washington Ave	Gotham	United States	wayneenterprises.com
1004	Aperture Laboratories	4545 Memorial Way	Ooahu	United States	aperturescience.com
1005	CBT Company	130 Advanced Drive	Sprinboro	United States	cbtcompany.com
1006	Allen-Bradley	124 Industrial Rd	Milwaukee	United States	allenbradley.com
1007	Scott Propulsion	4849 Enterprise Blvd	Glasgow	Scotland	scottprop.net
1008	Ishimura Mining	1890 Yamaha Rd	Kyoto	Japan	ishimuamine.com
1009	Stark Industries	985 Malibu Way	Malibu	United States	starkindustries.com
1010	Union Aerospace	333 Armstrong Ave	Houston	United States	unionaerospace.net
1011	IFM Efector	1671 Essen Way	Essen	Germany	ifm.com
1012	Rolls Royce	1221 Thames Blvd	London	United Kingdom	rollsroyce.com

POSITION_CODES TABLE

POSITION_CODE	POSITION
EE01	ElectricalEngineer
ME01	MechanicalEngineer
CE01	ControlsEngineer
ME02	MiningEngineer
OP01	Operator
EC02	Electrician
MC02	Mechanic
TC02	Technician
SD01	COBOLDeveloper
SD02	JavaDeveloper
QI01	Quality Inspector
SE01	SafetyEngineer

TASKS TABLE

TASKID	EQUIPMENTID	CATEGORY_CODE	CREATEDBY	ASSIGNEDTO	ASSIGNMENTDATE	FINISHEDDATE	TOTALHRS	COST	COMMENTS
6	WE001	CC001	13	23456	04-JAN-18	04-JAN-18	10	1000	Tumbler Computer needs to be reconfigured
7	WE001	IN001	13	10699	05-JAN-18	07-JAN-18	20	2000	Fire System needs to be replaced
8	SP223	IN002	10698	13	02-FEB-18	10-FEB-18	20	1000	Railings need to be placed around slip drive
9	AB001	RE001	13	992	15-APR-20	15-APR-20	5	30	VFD had overcurrent fault, bad bearings on spindle
10	WY102	CN001	12345	23456	09-JUL-20	10-JUL-20	10	47	Cold Intake to drive blocked
11	CS002	RE001	23456	(null)	20-NOV-20	(null)	(null)	(null)	(null)
12	UA001	(null)	(null)	(null)	20-NOV-20	(null)	(null)	(null)	Pressure Sensor on Driver HIGH Fault
13	IM002	RE002	(null)	(null)	20-NOV-20	(null)	(null)	(null)	(null)
14	WY103	RE001	12345	10699	07-SEP-20	08-SEP-20	100	1000	Drive Needs New Fusion Manifold
15	WE001	RE001	12345	12	09-SEP-20	10-SEP-20	100	10000	Tumbler Needs New Drive Axle

ITEM 3: Constraints

3a: Check Constraint – EQUIPMENT TABLE

CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE
EQUIPMENT_EQUIPMENTID_PK	Primary_Key	(null)	(null)	(null)	(null)	(null)
EQUIPMENT_MFGCODE_FK	Foreign_Key	(null)	CIS2268_FINAL	MANUFACTURERS	MANUFACTURERS_MFGCODE_PK	NO ACTION
EQUIPMENT_SERIALNO_UQ	Unique	(null)	(null)	(null)	(null)	(null)
EQUIPMENT_STATUS_CK	Check	status IN ('Active', 'Down')	(null)	(null)	(null)	(null)
SYS_C008318	Check	"SERIALNO" IS NOT NULL	(null)	(null)	(null)	(null)
SYS_C008319	Check	"PRODUCTNO" IS NOT NULL	(null)	(null)	(null)	(null)
SYS_C008320	Check	"LOCATION_CODE" IS NOT NULL	(null)	(null)	(null)	(null)

3b: Primary Key Constraint (multicolumn) – LOCATIONS TABLE

CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE
LOCATIONS_CODEFACILITY_PK	Primary_Key	(null)	(null)	(null)	(null)	(null)
SYS_C008338	Check	"FACILITY" IS NOT NULL	(null)	(null)	(null)	(null)
SYS_C008339	Check	"FLOOR" IS NOT NULL	(null)	(null)	(null)	(null)
SYS_C008340	Check	"ROOM" IS NOT NULL	(null)	(null)	(null)	(null)

```
/*##############################################CREATE LOCATIONS TABLE#############################################################################*/
CREATE TABLE locations
(   location_code CHARACTER (5),
    facility VARCHAR2 (10) NOT NULL,
    department VARCHAR2 (50),
    floor NUMBER (3) NOT NULL,
    room NUMBER (5) NOT NULL,
    description VARCHAR2 (50),

        CONSTRAINT locations_CodeFacility_pk PRIMARY KEY (location_code, facility));
```

3b: Primary Key Constraint (single column) – MANUFACTURERS TABLE

CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE
MANUFACTURERS_MFGCODE_PK	Primary_Key	(null)	(null)	(null)	(null)	(null)
MANUFACTURERS_MFG_UQ	Unique	(null)	(null)	(null)	(null)	(null)
SYS_C008304	Check	"MFG" IS NOT NULL	(null)	(null)	(null)	(null)
SYS_C008305	Check	"WEBSITE" IS NOT NULL	(null)	(null)	(null)	(null)

3c: Foreign Key Constraint – TASKS TABLE

CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE
SYS_C008342	Check	"ASSIGNMENTDATE" IS...	(null)	(null)	(null)	(null)
TASKS_ASSIGNEDTO_FK	Foreign_Key	(null)	CIS226...	EMPLOYEES	EMPLOYEES_EMPLO...	NO ACTION
TASKS_CATEGORYCODE_FK	Foreign_Key	(null)	CIS226...	TASK_CATEGORY	TASKCATEGORY_CA...	NO ACTION
TASKS_CREATEDBY_FK	Foreign_Key	(null)	CIS226...	EMPLOYEES	EMPLOYEES_EMPLO...	NO ACTION
TASKS_EQUIPMENTID_FK	Foreign_Key	(null)	CIS226...	EQUIPMENT	EQUIPMENT_EQUIP...	NO ACTION
TASKS_TASKID_PK	Primary_Key	(null)	(null)	(null)	(null)	(null)

3d: Unique / Not Null Constraint – EMPLOYEES TABLE

CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_OWNER	R_TABLE_NAME	R_CONSTRAINT_NAME	DELETE_RULE
EMPLOYEES_EMAIL_UQ	Unique	(null)	(null)	(null)	(null)	(null)
EMPLOYEES_EMPLOYEEID_PK	Primary_Key	(null)	(null)	(null)	(null)	(null)
EMPLOYEES_PHONENUMBER_UQ	Unique	(null)	(null)	(null)	(null)	(null)
EMPLOYEES_POSITIONCODE_FK	Foreign_Key	(null)	CIS226...	POSITION_C...	POSITIONCODES_P...	NO ACTION
SYS_C008311	Check	"FIRSTNAME" IS NOT ...	(null)	(null)	(null)	(null)
SYS_C008312	Check	"LASTNAME" IS NOT NULL	(null)	(null)	(null)	(null)
SYS_C008313	Check	"EMAIL" IS NOT NULL	(null)	(null)	(null)	(null)

ITEM 4: Queries from Multiple Tables

4a: Using Column Alias

```
/*ITEM 4a: USING COLUMN ALIAS*/
SELECT equipmentID, description AS Equipment_Name, mfg_code, mfg AS manufacturer
FROM manufacturers JOIN equipment USING (mfg_code);
```

EQUIPMENTID	EQUIPMENT_NAME	MFG_CODE	MANUFACTURER
WY103	WY Warp Drive 02	1000	Weyland-Yutani
WY102	WY Warp Drive 01	1000	Weyland-Yutani
CS002	Skynet Defense Sensor	1001	Cyberdyne Systems
WE001	WayneEnterprises Tumbler 01	1003	Wayne Industries
AB001	AB VFD PowerFlex 750	1006	Allen-Bradley
SP223	Scott Prop Slip Drive 1	1007	Scott Propulsion
IM002	Ishimua Rotary Drill	1008	Ishimura Mining
UA001	UnionAerospace Impact Driver	1010	Union Aerospace

4b: Performing Basic Arithmetic

```
/*ITEM 4b: PERFORMING BASIC ARITHMETIC*/
SELECT equipmentID, description, SUM(cost) AS Total_Costs, purchase_price, (purchase_price - SUM(cost)) AS Cost_to_Replace
FROM tasks JOIN equipment USING (equipmentID)
GROUP BY equipmentID, description, purchase_price;
```

EQU...	DESCRIPTION	TOTAL_COSTS	PURCHASE_PRICE	COST_TO_REPLACE
IM002	Ishimua Rotary Drill	(null)	(null)	(null)
SP223	Scott Prop Slip Drive 1	1000	9876543	9875543
CS002	Skynet Defense Sensor	(null)	14562	(null)
WY102	WY Warp Drive 01	46.72	1234567	1234520.28
AB001	AB VFD PowerFlex 750	3030.56	3000	-30.56
WY103	WY Warp Drive 02	1000.89	1234567	1233566.11
UA001	UnionAerospace Impact Driver	(null)	1122233	(null)
WE001	WayneEnterprises Tumbler 01	13001.57	1234561	1221559.43

4c: Removing Duplicates

```
SELECT DISTINCT productNo, mfg_code, mfg, website
FROM equipment JOIN manufacturers USING (mfg_code);
```

PRODUCTNO	MFG_CODE	MFG	WEBSITE
ISH-RD-002	1008	Ishimura Mining	ishimuamine.com
SNET1322	1001	Cyberdyne Systems	skynet.net
WE0011234	1003	Wayne Industries	wayneenterprises.com
WYWD134	1000	Weyland-Yutani	weylandyutani.com
AB-PF750-VFD-480	1006	Allen-Bradley	allenbradley.com
UAC1221ID	1010	Union Aerospace	unionaerospace.net
SP200	1007	Scott Propulsion	scottprop.net

4d: Using Concatenation

```
/*ITEM 4d: USING CONCATENATION*/
SELECT firstname || lastname, position_code, position
FROM employees JOIN position_codes USING (position_code);
```

FIRSTNAME||LASTNAME	POSITION_CODE	POSITION
HalKyles	SD01	COBOLDeveloper
YukiHamara	CE01	ControlsEngineer
DamonBaird	CE01	ControlsEngineer
IssacClarke	EE01	ElectricalEngineer
JimmyBuffett	SD02	JavaDeveloper
HomerHickam	ME01	MechanicalEngineer
AmandaRipley	ME01	MechanicalEngineer
CarterMatthews	ME02	MiningEngineer
SamirNadir	QI01	Quality Inspector
FrankJacobs	SE01	SafetyEngineer
WessBrookes	TC02	Technician

ITEM 5: Queries to Restrict Rows Using Criteria and Compound Criteria

5a: Range Criteria (between)

```
/*ITEM 5a: RANGE CRITERIA (between)*/
SELECT * FROM tasks
    WHERE cost BETWEEN 0 AND 1000;
```

TASKID	EQUIPMENTID	CATEGORY_CODE	CREATEDBY	ASSIGNEDTO	ASSIGNMENTDATE	FINISHEDDATE	TOTALHRS	COST
6	WE001	CC001	13	23456	04-JAN-18	04-JAN-18	10	1000
8	SP223	IN002	10698	13	02-FEB-18	10-FEB-18	20	1000
9	AB001	RE001	13	992	15-APR-20	15-APR-20	5	30
10	WY102	CN001	12345	23456	09-JUL-20	10-JUL-20	10	47
14	WY103	RE001	12345	10699	07-SEP-20	08-SEP-20	100	1000

5b: Using IN Operator

```
/*ITEM 5b: USING IN OPERATOR*/
SELECT * FROM tasks
    WHERE category_code IN ('RE001', 'RE002');
```

TASKID	EQUIPMENTID	CATEGORY_CODE	CREATEDBY	ASSIGNEDTO	ASSIGNMENTDATE	FINISHEDDATE	TOTALHRS	COST
9	AB001	RE001	13	992	15-APR-20	15-APR-20	5	30
11	CS002	RE001	23456	(null)	20-NOV-20	(null)	(null)	(null)
13	IM002	RE002	(null)	(null)	20-NOV-20	(null)	(null)	(null)
14	WY103	RE001	12345	10699	07-SEP-20	08-SEP-20	100	1000
15	WE001	RE001	12345	12	09-SEP-20	10-SEP-20	100	10000

5c: Using LIKE Operator with Wildcards

```
/*ITEM 5c: USING LIKE OPERATOR WITH WILDCARDS*/
SELECT * FROM equipment
    WHERE description LIKE 'WY%';
```

EQUIPMENTID	DESCRIPTION	STATUS	SERIALNO	PRODUCTNO	MFG_CODE	PURCHASE_PRICE	INSTALL_DATE	LOCATION_CODE
WY102	WY Warp Drive 01	Active	WY102-123-16	WYWD134	1000	1234567	08-OCT-19	PT001
WY103	WY Warp Drive 02	Active	WY102-123-17	WYWD134	1000	1234567	08-OCT-19	PT001

5d: Ordering of Output

```
/*ITEM 5d: Ordering of Output*/
SELECT * FROM tasks
    WHERE cost < 1000
    ORDER BY taskID DESC;

SELECT taskID, assignedTo, totalHrs, cost, comments FROM tasks
    WHERE assignedTo = 23456
    ORDER BY totalHrs;
```

TASKID	ASSIGNEDTO	TOTALHRS	COST	COMMENTS
10	23456	10	47	Cold Intake to drive blocked
6	23456	10	1000	Tumbler Computer needs to be reconfigured

ITEM 6: Joins

6a: Query that Joins 3 Tables and Limits Output

```
/*ITEM 6a: QUERY THAT JOINS 3 TABLES AND LIMITS OUTPUT*/
SELECT taskId, comments, equipmentID, description, mfg, assignedTo
FROM tasks JOIN equipment USING (equipmentID)
           JOIN manufacturers USING (mfg_code)
WHERE equipmentID = 'WE001';
```

TASKID	COMMENTS	EQUIPMENTID	DESCRIPTION	MFG	ASSIGNEDTO
6	Tumbler Computer needs to be reconfigured	WE001	WayneEnterprises Tumbler 01	Wayne Industries	23456
7	Fire System needs to be replaced	WE001	WayneEnterprises Tumbler 01	Wayne Industries	10699
15	Tumbler Needs New Drive Axle	WE001	WayneEnterprises Tumbler 01	Wayne Industries	12

6b: Query with an Outer Join

```
/*ITEM 6b: QUERY WITH AN OUTER JOIN*/
SELECT taskID, comments, equipmentID, description, mfg_code, mfg
FROM tasks LEFT OUTER JOIN equipment USING (equipmentID)
           JOIN manufacturers USING (mfg_code)
ORDER BY taskID;
```

TASKID	COMMENTS	EQUIPMENTID	DESCRIPTION	MFG_CODE	MFG
6	Tumbler Computer needs to be reconfigured	WE001	WayneEnterprises Tumbler 01	1003	Wayne Industries
7	Fire System needs to be replaced	WE001	WayneEnterprises Tumbler 01	1003	Wayne Industries
8	Railings need to be placed around slip drive	SP223	Scott Prop Slip Drive 1	1007	Scott Propulsion
9	VFD had overcurrent fault, bad bearings on spindle	AB001	Allen-Bradley VFD 750-01	1006	Allen-Bradley
10	Cold Intake to drive blocked	WY102	WY Warp Drive 01	1000	Weyland-Yutani
11	(null)	CS002	Skynet Defense Sensor	1001	Cyberdyne Systems
12	Pressure Sensor on Driver HIGH Fault	UA001	UnionAerospace Impact Driver	1010	Union Aerospace
13	(null)	IM002	Ishimua Rotary Drill	1008	Ishimura Mining

6c: Query with a Non-Equality Join

```
/*ITEM 6c: QUERY WITH A NON-EQUALITY JOIN*/
SELECT t.taskID, t.comments, t.cost, e.equipmentID, e.description, e.purchase_price
FROM tasks t, equipment e
WHERE t.cost < e.purchase_price
```

TASKID	COMMENTS	COST	EQUIPMENTID	DESCRIPTION	PURCHASE_PRICE
15	Tumbler Needs New Drive Axle	10000.56	SP223	Scott Prop Slip Drive 1	9876543
16	Gasket on Box Failed, water got into box	3000	SP223	Scott Prop Slip Drive 1	9876543
7	Fire System needs to be replaced	2000.2	SP223	Scott Prop Slip Drive 1	9876543
14	Drive Needs New Fusion Manifold	1000.89	SP223	Scott Prop Slip Drive 1	9876543
6	Tumbler Computer needs to be reconfigured	1000.81	SP223	Scott Prop Slip Drive 1	9876543
8	Railings need to be placed around slip drive	1000	SP223	Scott Prop Slip Drive 1	9876543
10	Cold Intake to drive blocked	46.72	SP223	Scott Prop Slip Drive 1	9876543
9	VFD had overcurrent fault, bad bearings on spindle	30.56	SP223	Scott Prop Slip Drive 1	9876543
15	Tumbler Needs New Drive Axle	10000.56	WY102	WY Warp Drive 01	1234567
16	Gasket on Box Failed, water got into box	3000	WY102	WY Warp Drive 01	1234567
7	Fire System needs to be replaced	2000.2	WY102	WY Warp Drive 01	1234567

6d: Query Using a Set Operation

```
/*ITEM 6d: QUERY USING A SET OPERATION*/
SELECT taskID, category_code, comments, equipmentID, description
FROM tasks JOIN equipment USING (equipmentID)
WHERE category_code = 'RE001'
UNION
SELECT taskID, category_code, comments, equipmentID, description
FROM tasks JOIN equipment USING (equipmentID)
WHERE category_code = 'RE002'
ORDER BY category_code
```

TASKID	CATEGORY_CODE	COMMENTS	EQUIPMENTID	DESCRIPTION
9	RE001	VFD had overcurrent fault, bad bearings on spindle	AB001	AB VFD PowerFlex 750
11	RE001	(null)	CS002	Skynet Defense Sensor
14	RE001	Drive Needs New Fusion Manifold	WY103	WY Warp Drive 02
15	RE001	Tumbler Needs New Drive Axle	WE001	WayneEnterprises Tumbler 01
17	RE001	Drill-Head Needs Replacement	IM002	Ishimua Rotary Drill
13	RE002	(null)	IM002	Ishimua Rotary Drill
16	RE002	Gasket on Box Failed, water got into box	AB001	AB VFD PowerFlex 750

ITEM 7: Single Row Functions

7a: Initcap

```
/*ITEM 7a: INITCAP*/
SELECT INITCAP(firstname), INITCAP(lastname) FROM employees;
```

INITCAP(FIRSTNAME)	INITCAP(LASTNAME)
Amanda	Ripley
Issac	Clarke
Damon	Baird
Carter	Matthews
Frank	Jacobs
Yuki	Hamara
Samir	Nadir
Hal	Kyles
Jimmy	Buffett

7b: Substrings

```
/*ITEM 7b: SUBSTRINGS*/
SELECT DISTINCT SUBSTR(equipmentID,1,2), description FROM equipment
ORDER BY SUBSTR(equipmentID,1,2);
```

SUBSTR(EQUIPMENTID,1,2)	DESCRIPTION
AB	Allen-Bradley VFD 750-01
CS	Skynet Defense Sensor
IM	Ishimua Rotary Drill
SP	Scott Prop Slip Drive 1
UA	UnionAerospace Impact Driver
WE	WayneEnterprises Tumbler 01
WY	WY Warp Drive 01
WY	WY Warp Drive 02

7c: Round

```
/*ITEM 7c: ROUND*/
SELECT taskID, equipmentID, comments, cost, ROUND(cost, 0) FROM tasks;
```

TASKID	EQUIPMENTID	COMMENTS	COST	ROUND(COST,0)
6	WE001	Tumbler Computer needs to be reconfigured	1000.81	1001
7	WE001	Fire System needs to be replaced	2000.2	2000
8	SP223	Railings need to be placed around slip drive	1000	1000
9	AB001	VFD had overcurrent fault, bad bearings on spindle	30.56	31
10	WY102	Cold Intake to drive blocked	46.72	47

7d: To Char – Use with Dates

```
/*ITEM 7d: TOCHAR - USE WITH DATES*/
SELECT equipmentID, description, TO_CHAR(install_date, 'MM-DD-YYYY') AS INSTALL_DATE
FROM equipment;
```

EQUIPMENTID	DESCRIPTION	INSTALL_DATE
WY102	WY Warp Drive 01	10-08-2019
WY103	WY Warp Drive 02	10-08-2019
CS002	Skynet Defense Sensor	02-03-2020
UA001	UnionAerospace Impact Driver	10-10-2019
WE001	WayneEnterprises Tumbler 01	10-31-2019
SP223	Scott Prop Slip Drive 1	12-12-2011
AB001	Allen-Bradley VFD 750-01	04-05-2015
IM002	Ishimua Rotary Drill	11-20-2020

7e: NVL

```
/*ITEM 7e: NVL*/
SELECT taskID, equipmentID, NVL(comments, 'Further Investigation Required'), NVL(totalhrs,1), cost
FROM tasks
WHERE NVL(comments, 'Further Investigation Required') = 'Further Investigation Required';
```

TASKID	EQUIPMENTID	NVL(COMMENTS,'FURTHERINVESTIGATIONREQUIRED')	NVL(TOTALHRS,1)	COST
11	CS002	Further Investigation Required	1	(null)
13	IM002	Further Investigation Required	1	(null)

7f: Cases

```
SELECT equipmentID, SUM(cost) AS MAINTENANCE_COST, purchase_price,
    CASE
        WHEN (SUM(cost) > purchase_price) THEN 'REPLACE'
        WHEN (SUM(cost) < purchase_price) THEN 'ACTIVE'
        ELSE 'CANNOT DETERMINE'
    END "Equip Life"
FROM tasks JOIN equipment USING(equipmentID)
GROUP BY equipmentID, purchase_price;
```

EQUIPMENTID	MAINTENANCE_COST	PURCHASE_PRICE	Equip Life
CS002	(null)	14562	CANNOT DETERMINE
SP223	1000	9876543	ACTIVE
WE001	13001.57	1234561	ACTIVE
WY103	1000.89	1234567	ACTIVE
AB001	3030.56	3000	REPLACE
IM002	(null)	(null)	CANNOT DETERMINE
WY102	46.72	1234567	ACTIVE
UA001	(null)	1122233	CANNOT DETERMINE

ITEM 8: Group Functions

8a: Sum and Average

```
/*ITEM 8a: SUM AND AVERAGE*/
SELECT equipmentID, equipment.description, SUM(cost) AS TOTAL_MAINTENANCE_COST,
ROUND(AVG(cost),2) AS AVG_MAINTENANCE_COST, purchase_price
FROM tasks JOIN equipment USING (equipmentID)
GROUP BY equipmentID, equipment.description, purchase_price;
```

EQUIPMENTID	DESCRIPTION	TOTAL_MAINTE...	AVG_MAIN...	PURCHASE_PRICE
IM002	Ishimua Rotary Drill	(null)	(null)	(null)
SP223	Scott Prop Slip Drive 1	1000	1000	9876543
CS002	Skynet Defense Sensor	(null)	(null)	14562
WY102	WY Warp Drive 01	46.72	46.72	1234567
WY103	WY Warp Drive 02	1000.89	1000.89	1234567
UA001	UnionAerospace Impact Driver	(null)	(null)	1122233
WE001	WayneEnterprises Tumbler 01	13001.57	4333.86	1234561
AB001	Allen-Bradley VFD 750-01	3030.56	1515.28	3000

8b: Count

```
/*ITEM 8b: COUNT*/
SELECT equipmentID, equipment.description, COUNT(equipmentID) AS MAINTENANCE_TASKS
FROM tasks JOIN equipment USING (equipmentID)
GROUP BY equipmentID, equipment.description;
```

EQUIPMENTID	DESCRIPTION	MAINTENANCE_TASKS
UA001	UnionAerospace Impact Driver	1
AB001	Allen-Bradley VFD 750-01	2
WE001	WayneEnterprises Tumbler 01	3
CS002	Skynet Defense Sensor	1
SP223	Scott Prop Slip Drive 1	1
WY103	WY Warp Drive 02	1
IM002	Ishimua Rotary Drill	1
WY102	WY Warp Drive 01	1

8c: Min and Max

```
/*ITEM 8c: MIN AND MAX FUNCTIONS*/
SELECT equipmentID, equipment.description, MIN(cost), MAX(cost), purchase_price
FROM tasks JOIN equipment USING (equipmentID)
GROUP BY equipmentID, equipment.description, equipment.purchase_price
```

EQUIPMENTID	DESCRIPTION	MIN(COST)	MAX(COST)	PURCHASE_PRICE
IM002	Ishimua Rotary Drill	(null)	(null)	(null)
SP223	Scott Prop Slip Drive 1	1000	1000	9876543
CS002	Skynet Defense Sensor	(null)	(null)	14562
WY102	WY Warp Drive 01	46.72	46.72	1234567
WY103	WY Warp Drive 02	1000.89	1000.89	1234567
UA001	UnionAerospace Impact Driver	(null)	(null)	1122233
WE001	WayneEnterprises Tumbler 01	1000.81	10000.56	1234561
AB001	Allen-Bradley VFD 750-01	30.56	3000	3000

8d: Group by with Single Table

```
/*ITEM 8d: GROUP WITH A SINGLE TABLE*/
SELECT COUNT(taskID) AS MAINTENANCE_COUNT, assignedTo, lastname
FROM tasks JOIN employees
    ON tasks.assignedTo = employees.employeeID
GROUP BY  assignedTo, lastname
```

MAINTENANCE_COUNT	ASSIGNEDTO	LASTNAME
2	992	Baird
2	10699	Hamara
1	13	Jacobs
1	12	Matthews
2	23456	Clarke

8e: Grouping Set or Cube

```
/*ITEM 8e: GROUP BY SET OR CUBE*/
SELECT COUNT(taskID), category_code, category, ROUND(AVG(cost),2) AS CAT_AVG_COST
FROM tasks JOIN task_category USING (category_code)
GROUP BY GROUPING SETS (category code, category, (category code, category));
```

COUNT(TASKID)	CATEGORY_CODE	CATEGORY	CAT_AVG_COST
4	RE001	REPLACEMENT - Part	3677.34
1	IN002	INSTALL - Large Part	1000
1	IN001	INSTALL - Small Part	2000.2
2	RE002	REPLACEMENT - Equipment	3000
1	CN001	CLEAN - General Cleaning	46.72
1	CC001	CONFIG-CHANGE - Config Change	1000.81
4	(null)	REPLACEMENT - Part	3677.34
1	(null)	INSTALL - Large Part	1000

ITEM 9: SUBQUERIES AND MERGE STATEMENTS

9a: Single-Row Subquery with a HAVING Clause

```
/*ITEM 9a: SINGLE-ROW SUBQUERY WITH A HAVING CLAUSE*/
SELECT employeeID, lastname, AVG(totalhrs)
FROM tasks JOIN employees
    ON tasks.assignedTo = employees.employeeID
HAVING SUM (totalhrs) > (SELECT AVG (totalhrs) FROM tasks)
GROUP BY employeeID, lastname;
```

EMPLOYEEID	LASTNAME	AVG(TOTALHRS)
10699	Hamara	60
12	Matthews	100

9b: Multi-Row Subquery with a HAVING Clause

```
/*ITEM 9b: MULTI-ROW SUBQUERY WITH HAVING CLAUSE*/
SELECT equipmentID, SUM(cost / totalhrs) AS Total_Maintenance
FROM tasks
HAVING SUM(cost / totalhrs) < ANY (SELECT SUM(purchase_price / totalhrs)
                                   FROM tasks JOIN equipment USING (equipmentID)
                                   GROUP BY equipmentID)
GROUP BY equipmentID;
```

EQUIPMENTID	TOTAL_MAINTENANCE
WY103	10.0089
SP223	50
WY102	4.672
AB001	156.112
WE001	300.0966

9c: Multi-Column Sub-Query with a Where Clause

```
/*ITEM 9c: MULTI-COLUMN SUBQUERY WITH A WHERE CLAUSE*/
SELECT category_code, category, taskID, cost AS HIGHEST_CAT_COST
FROM tasks JOIN task_category USING (category_code)
WHERE (category_code, cost) IN (SELECT category_code, MAX(cost)
                                FROM tasks
                                GROUP BY category_code)
ORDER BY category_code;
```

CATEGORY_CODE	CATEGORY	TASKID	HIGHEST_CAT_COST
CC001	CONFIG-CHANGE - Config Change	6	1000.81
CN001	CLEAN - General Cleaning	10	46.72
IN001	INSTALL - Small Part	7	2000.2
IN002	INSTALL - Large Part	8	1000
RE001	REPLACEMENT - Part	15	10000.56
RE002	REPLACEMENT - Equipment	16	3000

9d: Use a Subquery with the ALL Operator

```
/*TEM 9d: USE A SUBQUERY WITH ALL OPERATOR*/
SELECT taskID, comments, category_code, equipmentID, cost
FROM tasks
WHERE cost > ALL (SELECT cost
                  FROM tasks
                  WHERE category_code = 'CN001');
```

TASKID	COMMENTS	CATEGORY_CODE	EQUIPMENTID	COST
8	Railings need to be placed around slip drive	IN002	SP223	1000
6	Tumbler Computer needs to be reconfigured	CC001	WE001	1000.81
14	Drive Needs New Fusion Manifold	RE001	WY103	1000.89
7	Fire System needs to be replaced	IN001	WE001	2000.2
16	Gasket on Box Failed, water got into box	RE002	AB001	3000
15	Tumbler Needs New Drive Axle	RE001	WE001	10000.56

9e: Use a Sub-Query in a DML Statement

```
/*ITEM 9e: USE A SUBQUERY IN A DML ACTION*/
UPDATE tasks
SET assignedTo = (SELECT MAX(assignedTo)
                  FROM tasks)
WHERE assignedTo IS NULL;

SELECT * FROM tasks
```

TASKID	EQUIPMENTID	CA...	CREATEDBY	ASSIGNEDTO	ASSIGNME...	FINISHE...	TOT...	COST	COMMENTS
6	WE001	CC001	13	23456	04-JAN-18	04-JAN-18	10	1000.81	Tumbler Computer needs to be reconfigured
7	WE001	IN001	13	10699	05-JAN-18	07-JAN-18	20	2000.2	Fire System needs to be replaced
8	SP223	IN002	10698	13	02-FEB-18	10-FEB-18	20	1000	Railings need to be placed around slip drive
9	AB001	RE001	13	992	15-APR-20	15-APR-20	5	30.56	VFD had overcurrent fault, bad bearings on spindl
10	WY102	CN001	12345	23456	09-JUL-20	10-JUL-20	10	46.72	Cold Intake to drive blocked
11	CS002	RE001	23456	23456	21-NOV-20	(null)	(null)	(null)	(null)
12	UA001	(n...	(null)	23456	21-NOV-20	(null)	(null)	(null)	Pressure Sensor on Driver HIGH Fault
13	IM002	RE002	(null)	23456	21-NOV-20	(null)	(null)	(null)	(null)
14	WY103	RE001	12345	10699	07-SEP-20	08-SEP-20	100	1000.89	Drive Needs New Fusion Manifold
15	WE001	RE001	12345	12	09-SEP-20	10-SEP-20	100	10000.56	Tumbler Needs New Drive Axle
16	AB001	RE002	12345	992	10-OCT-20	11-OCT-20	20	3000	Gasket on Box Failed, water got into box
17	IM002	RE001	(null)	23456	24-NOV-20	(null)	(null)	(null)	Drill-Head Needs Replacement

ITEM 11: Views

11a: Create a Simple View

```
/*ITEM 11a: CREATE A SIMPLE VIEW*/
CREATE OR REPLACE VIEW OPEN_TASKS
AS SELECT taskID, comments, equipmentID, equipment.description AS Equipment,
locations.description AS Location, createdBy, assignedTo, totalhrs, cost
FROM tasks JOIN equipment USING (equipmentID)
           JOIN locations USING (location_code)
WHERE finishedDate IS NULL;
SELECT * FROM OPEN_TASKS
```

	TASKID	COMMENTS	EQUIPMENTID	EQUIPMENT	LOCATION	CREATEDBY	ASSIGNEDTO	TOTALHRS	COST
1	12	Pressure Sensor on Driver HIGH Fault	UA001	UnionAerospace Impact Driver	Eng Lab 2	(null)	23456	(null)	(null)
2	13	(null)	IM002	Ishimua Rotary Drill	Eng Lab 2	(null)	23456	(null)	(null)
3	17	Drill-Head Needs Replacement	IM002	Ishimua Rotary Drill	Eng Lab 2	(null)	23456	(null)	(null)
4	11	(null)	CS002	Skynet Defense Sensor	AI Dev Lab 10	23456	23456	(null)	(null)

11b: Create a View with a Check Option

```
/*ITEM 11b: CREATE A VIEW WITH A CHECK OPTION*/
CREATE OR REPLACE VIEW CLOSED_TASKS AS
SELECT taskID, comments, equipmentID, description, finishedDate, totalhrs, cost, lastname AS CompletedBy
FROM tasks JOIN equipment USING (equipmentID)
           JOIN employees
ON tasks.assignedTo = employees.employeeID
WHERE finishedDate IS NOT NULL
WITH CHECK OPTION
```

TASKID	COMMENTS	EQUIPMENTID	DESCRIPTION	FINISHEDDATE	TOTALHRS	COST	COMPLETEDBY
10	Cold Intake to drive blocked	WY102	WY Warp Drive 01	10-JUL-20	10	46.72	Clarke
6	Tumbler Computer needs to be reconfigured	WE001	WayneEnterprises Tumbler 01	04-JAN-18	10	1000.81	Clarke
16	Gasket on Box Failed, water got into box	AB001	Allen-Bradley VFD 750-01	11-OCT-20	20	3000	Baird
9	VFD had overcurrent fault, bad bearings on spindle	AB001	Allen-Bradley VFD 750-01	15-APR-20	5	30.56	Baird
15	Tumbler Needs New Drive Axle	WE001	WayneEnterprises Tumbler 01	10-SEP-20	100	10000.56	Matthews
8	Railings need to be placed around slip drive	SP223	Scott Prop Slip Drive 1	10-FEB-18	20	1000	Jacobs
14	Drive Needs New Fusion Manifold	WY103	WY Warp Drive 02	08-SEP-20	100	1000.89	Hamara
7	Fire System needs to be replaced	WE001	WayneEnterprises Tumbler 01	07-JAN-18	20	2000.2	Hamara

11c: Update a Record in a Simple View

```
/*ITEM 11c: UPDATE A RECORD IN A SIMPLE VIEW*/
UPDATE EQUIPMENT_IDENTIFICATION
    SET description = 'AB VFD PowerFlex 750'
    WHERE description = 'Allen-Bradley VFD 750-01'
```

EQUIPMENTID	DESCRIPTION	MFG_CODE	MFG	SERIALNO	PRODUCTNO
AB001	AB VFD PowerFlex 750	1006	Allen-Bradley	PW-750-00123	AB-PF750-VFD-480
CS002	Skynet Defense Sensor	1001	Cyberdyne Systems	CS-SKY-01241	SNET1322
IM002	Ishimua Rotary Drill	1008	Ishimura Mining	ISH297-312	ISH-RD-002

11d: Perform TOP-N Analysis

```
/*ITEM 11d: TOP-N ANALYSIS*/
SELECT equipmentID, description, purchase_price
FROM (SELECT equipmentID, description, purchase_price
        FROM equipment
        WHERE purchase_price IS NOT NULL
        ORDER BY purchase_price DESC)
WHERE ROWNUM <=5;
```

EQUIPMENTID	DESCRIPTION	PURCHASE_PRICE
SP223	Scott Prop Slip Drive 1	9876543
WY102	WY Warp Drive 01	1234567
WY103	WY Warp Drive 02	1234567
WE001	WayneEnterprises Tumbler 01	1234561
UA001	UnionAerospace Impact Driver	1122233

ITEM 12: Sequences and Indexes

12a: Create a Sequence and Use it When Inserting Records

```
/*ITEM 12a: CREATE A SEQUENCE*/
CREATE SEQUENCE task_taskID_seq
INCREMENT BY 1
START WITH 00017
NOCYCLE

INSERT INTO tasks(taskID, equipmentID, category_code, comments)
VALUES (task_taskID_seq.NEXTVAL, 'IM002', 'RE001', 'Drill-Head Needs Replacement');

SELECT * FROM tasks;
```

14	WY103	RE001	12345	10699	07-SEP-20	08-SEP-20	100	1000.89	Drive Needs New Fusion Manifold
15	WE001	RE001	12345	12	09-SEP-20	10-SEP-20	100	10000.56	Tumbler Needs New Drive Axle
16	AB001	RE002	12345	992	10-OCT-20	11-OCT-20	20	3000	Gasket on Box Failed, water got into bo:
17	IM002	RE001	(null)	(null)	24-NOV-20	(null)	(null)	(null)	Drill-Head Needs Replacement

12b: List All Sequences Using a Query

```
/*ITEM 12b: LIST ALL SEQUENCES USING A QUERY*/
SELECT * FROM user_sequences;
```

SEQUENCE_NAME	MIN_VALUE	MAX_VALUE	INCREMENT_BY	CYCLE_FLAG	ORDER_FLAG	CACHE_SIZE	LAST_NUMBER
ADD_TASK	1	9999999999999999999999999999	1	N	N	20	17
DEMO_CUST_SEQ	1	9999999999999999999999999999	1	N	N	20	21
DEMO_ORDER_ITEMS_SEQ	1	9999999999999999999999999999	1	N	N	20	61
DEMO_ORD_SEQ	1	9999999999999999999999999999	1	N	N	20	11
DEMO_PROD_SEQ	1	9999999999999999999999999999	1	N	N	20	21
DEMO_USERS_SEQ	1	9999999999999999999999999999	1	N	N	20	21
TASK_TASKID_SEQ	1	9999999999999999999999999999	1	N	N	20	37

12c: Create an Index (other than a primary key)

```
/*ITEM 12c: CREATE AN INDEX (OTHER THAN UNIQUE OR PRIMARY KEY*/
CREATE INDEX manufacturer_mfgCode_idx
ON manufacturers (mfg_code, mfg);

SELECT table_name, index_name, index_type
FROM user_indexes WHERE table_name = 'MANUFACTURERS';
```

TABLE_NAME	INDEX_NAME	INDEX_TYPE
MANUFACTURERS	MANUFACTURERS_MFGCODE_PK	NORMAL
MANUFACTURERS	MANUFACTURERS_MFG_UQ	NORMAL
MANUFACTURERS	MANUFACTURER MFGCODE IDX	NORMAL

12d: List All Indexes Using a Query

```
/*ITEM 12d: LIST ALL INDEXES USING A QUERY*/
SELECT table_name, index_name, index_type
FROM user_indexes;
```

MANUFACTURERS	MANUFACTURERS_MFGCODE_PK	NORMAL
DEMO_CUSTOMERS	DEMO_CUSTOMERS_PK	NORMAL
DEMO_CUSTOMERS	DEMO_CUST_NAME_IX	NORMAL
DEMO_ORDERS	DEMO_ORDER_PK	NORMAL
DEMO_ORDERS	DEMO_ORD_CUSTOMER_IX	NORMAL
LOCATIONS	LOCATIONS_CODEFACILITY_PK	NORMAL
TASKS	TASKS_TASKID_PK	NORMAL
MANUFACTURERS	MANUFACTURER_MFGCODE_IDX	NORMAL

12e: Identify a Column You Would Create a Bitmap Index for and Why

One column that could be selected for a bitmap index is the 'category_code' column in the TASKS table. Bitmap indexes are usually selected for columns that have a low number of distinct values. Currently, the 'category_code' column is link to the TASK_CATEGORY table, where there are only 6 records.

Maintenance tasks would use the 'category_code' to help describe the maintenance that was performed. This column would not expand much further, since the 'category_code' column should be a broad description of maintenance performed. This column should not be used to provide specific maintenance task information.

```
CREATE BITMAP INDEX tasks_taskCategory_idx
ON tasks (category_code)
```

(creating bitmap index was 'commented-out' in script)

ITEM 13: Write a Security Plan for your Database

13a: What User Would You Create?

For each employee in the EMPLOYEES table, a user would need to be created. This would allow the database administrator to audit and track user's changes and modifications. If a user made an undesired change, they could be questioned. Each user in the EMPLOYEES table would be allowed to create new tasks in the TASKS table.

13b: What Roles Would Need to be Created?

To accommodate the different jobs/roles used throughout the organization, different database roles would be created. This follows the principle of least privilege, ensuring that users have the bare minimum privileges to perform their job. The roles that would be created are as follows:

- OPERATORS
- MANAGERS

- ENGINEERS
- HR EMPLOYEES
- DATABASE ADMIN

13c: What Privileges would you Grant?

OPERATORS: The operators would have the bare minimum set of privileges from the discussed roles. The operators would be responsible for updating records within the TASKS table. The Operators would be assigned to employees that would be performing maintenance tasks and would be interacting with the table on a daily basis. However, they would not be interacting with other tables, they would not be concerned with editing records in tables such as EQUIPMENT, MANUFACTURERS, POSITION_CODES.

SYSTEM PRIVILEGES

- CREATE SESSION
- SELECT ANY TABLE (to look at all equipment, mfg, employee information, cannot make modifications to tables other than TASKS)

OBJECT PRIVILEGES

- UPDATE tasks
- INSERT tasks

MANAGERS / ENGINEERS: Managers and Engineers would have the same privileges as OPERATORS (ability to edit records within the TASKS table), but with additional privileges to edit records from EQUIPMENT, MANUFACTURERS, and LOCATIONS. Occasionally, records in these tables might need to be changed, management personal should be the only allowed personnel to do so. This is mainly because management would be responsible for interacting with manufacturers, ordering equipment, and possible installation of equipment.

SYSTEM PRIVILEGES

- CREATE SESSION
- SELECT ANY TABLE

OBJECT PRIVILEGES

- INSERT tasks, equipment, manufacturers, location, task_category
- UPDATE tasks, equipment, manufacturers, locations, task_category

HR EMPLOYEES: The HR EMPLOYEES role would have the ability to make modifications to the POSITION_CODES, and EMPLOYEES table. This is mainly to ensure that personnel records are kept consistent with any other organizational documents. The HR EMPLOYEES role would be the only role permitted to make modifications to these two tables.

SYSTEM PRIVILEGES

- CREATE SESSION

OBJECT PRIVILEGES

- INSERT employees, position_codes
- UPDATE employees, position_codes

DATABASE ADMIN: The DATABASE ADMIN role would be the system administrator; they would have complete control over the database and be allowed to perform any action. This user would only interact with the database on special occasions, like when a new table needs to be created, the database becomes corrupted and backups need to be restored.

The MANAGERS/ENGINEERS should not have complete control over the database to help prevent corruption and misuse. They are only given privileges to make changes to tables that they interact with. Those changes are only to update and insert new records. The MANAGERS/ENGINEERS cannot alter the underlying structure of the database, that privilege is only granted to the DATABASE ADMIN

SYSTEM PRIVILEGES (GIVEN DBA ROLE)

- ADMIN