```
                                        ON pr.PropertyID = wo2.PropertyID
                                            WHERE DATEDIFF(YEAR, wo2.LastMaintDate,
GETDATE()) <= 1);

/* 7. A listing of all work orders by status. */
SELECT *
        FROM WORK_ORDERS
ORDER BY WorkOrderStatus;

/* 8. A listing of technicians and the number of new work orders in the past month - April */
SELECT TechID, CONCAT(EmpFName, ' ',EmpLName) as "Name",
                COUNT(WorkOrderID) AS "Count of New Work Orders in the past month"
        FROM TECHNICIANS t JOIN WORK_ORDERS w
                 ON w.TechID = t.EmpID
                JOIN EMPLOYEES e
                 ON t.EmpID = e.EmpID
        --BETWEEN is used because there is sample data with dates in Oct 2020
        WHERE DATEDIFF(MONTH, DesiredWorkDate, GETDATE()) BETWEEN 0 AND 1
                GROUP BY TechID, CONCAT(EmpFName, ' ',EmpLName);

/* Advanced Test #9 - Views (Taken from the "RESULTS" section of Milestone 2)
9. "Residents who have had maintenance work completed in the past 6 months." */
CREATE VIEW ResidentsRecentMaintenance AS
        (SELECT R.ResidentID AS "Resident ID",
             R.ResApt AS "Resident's Apartment #",
                R.ResName AS "Resident's Name",
                W.WorkOrderID AS "Work Order ID",
                W.LastMaintDate AS "Date Last Maintenance Occurred"
            FROM RESIDENTS R JOIN WORK_ORDERS W
             ON R.ResidentID = W.ResidentID
                 WHERE DATEDIFF(MONTH,W.LastMaintDate,GETDATE()) <= 6);
SELECT *
        FROM ResidentsRecentMaintenance;

/* 10. Number of work orders each employee created. */
SELECT EmpID, CONCAT(EmpFName, ' ',EmpLName) as "Name",
                COUNT(CreatorID) as "Number of Work Orders Each Employee Created"
        FROM WORK_ORDERS w JOIN EMPLOYEES e
                ON e.EmpID = w.CreatorID
        GROUP BY EmpID, CONCAT(EmpFName, ' ',EmpLName)
ORDER BY 3 desc;
```

```
/* 11. Accident report on all employees. */
SELECT CONCAT(EmpFName, ' ',EmpLName) as "Name",
                a.AccidentType, a.AccID, a.OccurenceDate, a.ResumeDate, a.Status
        FROM ACCIDENTS a JOIN EMPLOYEES e
                ON e.EmpID = a.EmpID

/*Advanced Query #8
12. Total number of accidents per month for the past year by accident status. */
create view MonthsWithAccidents as
   select DATENAME(month, OccurenceDate) as "Months with Accidents",
          COUNT(DATENAME(month, OccurenceDate)) as "Number of Accidents",
          Status
        from ACCIDENTS
          where OccurenceDate > DATEADD(year,-1,GETDATE())
                group by DATENAME(month, OccurenceDate), Status;

/* 13. Listing of all properties and residents.*/
SELECT r.ResName as Name, r.ResApt as 'Apt Num',
                p.PropertyAddress, p.PropertyCity, p.PropertyState, p.PropertyName,
p.PropertyAccountNo
        FROM RESIDENTS r JOIN WORK_ORDERS w
                  ON w.ResidentID = r.ResidentID
                JOIN PROPERTY p
                  ON p.PropertyID = w.PropertyID

/* 14. A listing of users and the number of days the account was active. */
SELECT u.UserID,
          -- If the value below is not null, this means the users' credentials has been retired.
          DATEDIFF(DAY, u.DateSetUpStamp, u.DateRetired) AS "Number of Days Account was Active If
Retired",
          -- If the value above is null, then the client may use this value for active users.
          DATEDIFF(DAY, u.DateSetUpStamp, GETDATE()) AS "Number of Days Account Would Be Active
If Not Retired"
                FROM USERS u;

/* 15. Suppliers who have defaulted on a delivery. */
SELECT *
        FROM SUPPLIERS
WHERE DeliveryRating = 'Defaulted'
```

SQL scripts of all tests, transactions, adv. queries, triggers/procedures, optimization, views (all scripts will also be included in a notepad file (.txt) and submitted with this report)

Integrity Tests

```
/* Entity integrity tests- Initials: JL, MB, IS, VS */
/* Entity integrity test for EMPLOYEES */
-- Insert scripts to test uniqueness.
Insert into EMPLOYEES values ('0123456789', 'Jack', 'Smith', '1234 West Adams Street, Phoenix, AZ 99921', '333-345-5555', 'T', 'Jane Smith', 'Mother', '222-222-2222');
Insert into EMPLOYEES values ('4359307467', 'Jusuf', 'Nurkic', '90000 East Beaverton Lane, Portland, OR 78902', '802-789-3456', 'T', 'Hariz Nurkic', 'Father', '424-392-5307');
Insert into EMPLOYEES values ('3781003956', 'Jimmy', 'Butler', '12345 Chicago Boulevard, Chicago, IL, 34231', '710-241-6433', 'T', 'Rashon Butler', 'Father', '734-718-4418');
-- Selects all 60 primary keys to show they are unique.
SELECT DISTINCT E.EmpID AS 'Employee ID'
        FROM EMPLOYEES E
                -- Proves all 60 primary keys do not have NULL values.
                WHERE E.EmpID IS NOT NULL;

/* Entity integrity test for Accidents */
-- Selects all 30 primary keys to prove all 30 primary keys are unique/distinct.
SELECT DISTINCT A.AccID AS 'Accident ID',
          A.AccidentAddress AS 'Accident Address',
          A.EmpID AS 'ID of Employee Involved'
        FROM ACCIDENTS A
                -- Proves all 30 primary keys do not have NULL values.
                WHERE A.AccID IS NOT NULL;

/* Entity integrity for TECHNICIANS table */
-- Selects 30 distinct primary keys to prove all primary keys are unique.
SELECT DISTINCT T.EmpID AS 'Technician ID',
          T.TradeSpecialty AS 'Trade Specialty'
        FROM TECHNICIANS T
                WHERE T.EmpID IS NOT NULL;

/* Entity integrity for OFFICE_STAFF table */
-- Selects all 30 distinct primary keys to prove all primary keys in the table are unique.
SELECT DISTINCT O.EmpID AS 'Office Staff ID',
          O.HrsWorkedPerWeek AS 'Hours Worked Per Week',
          O.HourlyRate AS 'Hourly Pay Rate'
```

```
        FROM OFFICE_STAFF O
                --Proves all 30 distinct primary keys do not have NULL values in them.
                WHERE O.EmpID IS NOT NULL;

/* Entity integrity test for WORK_ORDERS table */
-- Selects all 30 distinct primary keys to prove all primary keys are unique.
SELECT DISTINCT W.WorkOrderID AS 'Work Order ID',
          W.DateCreated AS 'Date of Work Order Creation',
          W.DesiredWorkDate AS 'Desired Date to Start Work'
        FROM WORK_ORDERS W
                -- Proves all of the 30 primary keys do not have NULL values.
                WHERE W.WorkOrderID IS NOT NULL;

/* Entity integrity test for RESIDENTS table */
-- Selects all 30 distinct primary keys to prove all primary keys are unique.
SELECT DISTINCT R.ResidentID AS 'Resident ID',
          R.ResName AS 'Resident Name',
          R.ResApt AS 'Resident Apartment Number'
                FROM RESIDENTS R
                        -- Ensures all 30 primary keys do not have null values in them.
                        WHERE R.ResidentID IS NOT NULL;

/* Entity integrity test for TASKS. */
-- Selects all 30 distinct primary to keys to prove their uniqueness.
SELECT DISTINCT T.TaskID AS 'Task ID',
          T.TaskType AS 'Task Type',
          T.TaskDescription AS 'Task Description'
        FROM TASKS T
                -- Proves all 30 primary keys do not have null values.
                WHERE T.TaskID IS NOT NULL;

/* Entity integrity test for PROPERTY table */
-- Selects all 30 distinct primary keys to prove all primary keys are unique.
SELECT DISTINCT P.PropertyID AS 'Property ID',
          P.PropertyName AS 'Property Name',
          P.PropertyPhone AS 'Property Phone Number'
                FROM PROPERTY P
                        -- Ensures all 30 primary keys do not have null values in them.
                        WHERE P.PropertyID IS NOT NULL;

/* Entity integrity test for ASSETS table */
-- Selects all 30 distinct primary keys to prove all primary keys are unique.
```

```
SELECT DISTINCT A.AssetID AS 'Asset ID',
            A.AssetType AS 'Asset Type',
            A.AssetDescript AS 'Description'
                  FROM ASSETS A
                        -- Ensures all 30 primary keys do not have null values in them.
                        WHERE A.AssetID IS NOT NULL;

/* Entity integrity test for PARTS_PURCHASED table */
-- Insert Scripts to test uniqueness
Insert into PARTS_PURCHASED values('8850646215', '1553882107');
Insert into PARTS_PURCHASED values('1204535835', '1553882107');
-- Selects all 60 distinct primary keys/foreign keys to prove that they are unique
SELECT DISTINCT PP.PartID AS 'Part ID',
            PP.SupplierID AS 'Supplier ID'
                  FROM PARTS_PURCHASED PP
                        -- Ensures all 60 primary keys do not have null values in them.
                        WHERE PP.PartID IS NOT NULL
                        and PP.SupplierID IS NOT NULL;

/* Entity integrity test for PARTS_USED table */
-- Insert Scripts to test uniqueness
Insert into PARTS_USED values ('2393881879', '1204535835');
Insert into PARTS_USED values ('2393881879', '3048021228');
-- Selects all 60 distinct primary keys/foreign keys to prove that they are unique
SELECT DISTINCT PU.PartID AS 'Part ID',
            PU.WorkOrderID AS 'Work Order ID'
                  FROM PARTS_USED PU
                        -- Ensures all 60 primary keys do not have null values in them.
                        WHERE PU.PartID IS NOT NULL
                        and PU.WorkOrderID IS NOT NULL;

/* Entity integrity test for ASSET_SUPPLIER table */
-- Insert Scripts to test uniqueness
Insert into ASSET_SUPPLIER values('1553882107', '2874624385');
Insert into ASSET_SUPPLIER values('1553882107', '0030025687');
-- Selects all 30 distinct primary keys to prove all primary keys are unique.
SELECT DISTINCT A.AssetID AS 'Asset ID',
            A.SupplierID AS 'Supplier ID'
                  FROM ASSET_SUPPLIER A
                        -- Ensures all 30 primary keys do not have null values in them.
                        WHERE A.AssetID IS NOT NULL
                        and A.SupplierID IS NOT NULL;
```

```
/* Entity integrity test for USER table */
-- Insert Scripts to test uniqueness
Insert into USERS values ('8931072793', '7944061124', 'F-4g85bHPW', '2016-05-12', Null);
Insert into USERS values ('0341388010', '5425516460', 'Ir4rbBEj+0_A7WL', '2010-09-12', Null);
-- Selects all 30 distinct primary keys to prove all primary keys are unique.
SELECT DISTINCT U.UserID AS 'User ID',
            U.EmpID AS 'Employee ID',
            U.UserPassword AS 'Password'
                  FROM USERS U
                        -- Ensures all 30 primary keys do not have null values in them.
                        WHERE U.UserID IS NOT NULL;

/* Referential integrity tests - Initials: VS, MB, IS */

/* Referential integrity test for ACCIDENTS and EMPLOYEES relationship */
-- Displays 30 rows of data, thereby validating referential integrity.
-- Selects attributes from both tables to show referential integrity.
SELECT A.AccID AS 'Accident ID',
             E.EmpFName + ' ' + E.EmpLName AS 'Name of Employee Involved'
           -- Joins together tables on primary key/foreign key to validate referential integrity.
           FROM ACCIDENTS A JOIN EMPLOYEES E
            ON E.EmpID = A.EmpID

/* Referential integrity test for EMPLOYEES AND TECHNICIANS relationships */
-- Displays 30 rows of data, thereby validating referential integrity.
-- Selects attributes from both tables to show referential integrity.
SELECT T.EmpID AS 'Technician ID',
             E.EmpFName AS 'Technician First Name',
             E.EmpLName AS 'Technician Last Name',
             T.TradeSpecialty AS 'Technician Specialty'
           -- Joins together tables on primary key/foreign key to validate referential integrity.
           FROM TECHNICIANS T JOIN EMPLOYEES E
            ON T.EmpID = E.EmpID;

/* Referential integrity test for EMPLOYEES AND OFFICE_STAFF relationships */
-- Displays 30 rows of data, thereby validating referential integrity.
-- Selects attributes from both tables to show referential integrity.
SELECT O.EmpID AS 'Office Staff ID',
             E.EmpFName AS 'Office Staff First Name',
             E.EmpLName AS 'Office Staff Last Name',
             O.HrsWorkedPerWeek AS 'Hours Worked Per Week'
```

```
        -- Joins together tables on primary key/foreign key to validate referential integrity.
        FROM OFFICE_STAFF O JOIN EMPLOYEES E
          ON O.EmpID = E.EmpID;

/* Referential integrity test for relationships between WORK_ORDERS AND the following tables:
TECHNICIANS, OFFICE_STAFF, RESIDENTS, PROPERTY, TASKS */
-- Displays 30 rows of data, thereby validating referential integrity.
-- Selects attributes from all tables to validate referential integrity.
SELECT W.WorkOrderID AS 'Work Order ID',
          T.TradeSpecialty AS 'Technician Specialty',
          O.HrsWorkedPerWeek AS 'Office Staff Hrs Per Week',
          R.ResName AS 'Resident Name',
          P.PropertyName AS 'Property Name',
          TS.TaskType AS 'Task Type'
        -- Joins together tables on primary key/foreign key to validate referential integrity.
        FROM WORK_ORDERS W, TECHNICIANS T, OFFICE_STAFF O, RESIDENTS R, PROPERTY P, TASKS
TS
                WHERE W.TechID = T.EmpID
                  AND W.CreatorID = O.EmpID
                  AND W.ResidentID = R.ResidentID
                  AND W.PropertyID = P.PropertyID
                  AND W.TaskID = TS.TaskID;

/* Referential integrity test for the relationship between PROPERTY and ASSETS */
-- Displays 30 rows of data, thereby validating referential integrity.
-- Selects attributes from both tables to validate referential integrity.
SELECT P.PropertyName AS 'Property Name',
           A.AssetDescript AS 'Asset Description',
           A.AssetType AS 'Asset Type'
        -- Joins together tables on primary key/foreign key to validate referential integrity.
        FROM ASSETS A JOIN PROPERTY P
          ON A.PropertyID = P.PropertyID;

/* Referential integrity test for the relationship between WORK_ORDERS AND PARTS_USED */
-- Displays 60 rows of data, thereby validating referential integrity.
-- Selects attributes from both tables to validate referential integrity.
SELECT W.LastMaintDate AS 'Work Order Last Maintenance Date',
           PU.WorkOrderID AS 'Work Order ID',
           PU.PartID AS 'Part ID'
        -- Joins together tables on primary key/foreign key to validate referential integrity.
        FROM WORK_ORDERS W JOIN PARTS_USED PU
          ON W.WorkOrderID = PU.WorkOrderID;
```

```
/* Referential integrity test for the relationship between PARTS_USED AND PARTS */
-- Displays 60 rows of data, thereby validating referential integrity.
-- Selects attributes from both tables to validate referential integrity.
SELECT PU.WorkOrderID AS 'Work Order ID',
            PU.PartID AS 'Part ID',
            P.PartCategory AS 'Part Category'
          -- Joins together tables on primary key/foreign key to validate referential integrity.
          FROM PARTS P JOIN PARTS_USED PU
           ON P.PartID = PU.PartID;

/* Referential integrity test for the relationship between PARTS_PURCHASED AND PARTS */
-- Displays 60 rows of data, thereby validating referential integrity.
-- Selects attributes from both tables to validate referential integrity.
SELECT PP.SupplierID AS 'Supplier ID',
            PP.PartID AS 'Part ID',
            P.PartCategory AS 'Part Category',
            P.PartSubCategory AS 'Part Subcategory'
          -- Joins together tables on primary key/foreign key to validate referential integrity.
          FROM PARTS P JOIN PARTS_PURCHASED PP
           ON P.PartID = PP.PartID;

/* Referential integrity test for the relationship between PARTS_PURCHASED AND SUPPLIERS */
-- Displays 60 rows of data, thereby validating referential integrity.
-- Selects attributes from both tables to validate referential integrity.
SELECT PP.PartID AS 'Part ID',
            PP.SupplierID AS 'Supplier ID',
            S.SupplierName AS 'Supplier Name'
          -- Joins together tables on primary key/foreign key to validate referential integrity.
          FROM PARTS_PURCHASED PP JOIN SUPPLIERS S
           ON PP.SupplierID = S.SupplierID;

/* Referential integrity test for the relationship between SUPPLIERS AND ASSET_SUPPLIER */
-- Displays 60 rows of data, thereby validating referential integrity.
-- Selects attributes from both tables to validate referential integrity.
SELECT A.AssetID AS 'Asset ID',
            A.SupplierID AS 'Supplier ID',
            S.SupplierName AS 'Supplier Name'
          -- Joins together tables on primary key/foreign key to validate referential integrity.
          FROM SUPPLIERS S JOIN ASSET_SUPPLIER A
           ON S.SupplierID = A.SupplierID;
```

```
/* Referential integrity test for the relationship between ASSET_SUPPLIER AND ASSET */
-- Displays 60 rows of data, thereby validating referential integrity.
-- Selects attributes from both tables to validate referential integrity.
SELECT ASU.SupplierID AS 'Supplier ID',
            ASU.AssetID AS 'Asset ID',
            A.AssetDescript AS 'Asset Description',
            A.AssetType AS 'Asset Type'
          -- Joins together tables on primary key/foreign key to validate referential integrity.
          FROM ASSETS A JOIN ASSET_SUPPLIER ASU
           ON A.AssetID = ASU.AssetID;
```

Basic and Advanced Query Tests

```
* BASIC QUERIES */

/*Basic Test #1. Two INSERT statements executed together – the first one adding a new Employee, the
second one
adding a new Accident involving that Employee, should succeed.*/

/* BEFORE */
SELECT *
   FROM ACCIDENTS;
SELECT *
   FROM EMPLOYEES;

Insert into EMPLOYEES
        values ('2346969567', 'Jones', 'MF', '1234 East Adams Street, Phoenix, AZ 99921', '333-345-
5555', 'T', 'Jane Smith', 'Mother', '222-222-2222');
Insert into ACCIDENTS
        values ('2343636969', '2346969567', '11-07-2018', '333 North Scottsdale Road, Scottsdale, AZ,
85260', 'Fall from ladder', '11-08-2018', '01-07-2019','8883412745','Major');

/* AFTER */
select *
   from ACCIDENTS
        where EmpID = '2346969567';
select *
   from EMPLOYEES
        where EmpID = '2346969567';

/*Basic Test #2. A similar two INSERT statements executed together in the opposite order should fail. */
```

```
/* BEFORE */
SELECT *
   FROM EMPLOYEES
         WHERE EmpID = '0222269813';

Insert into ACCIDENTS
         values ('9356683562', '0222269813', '09-16-2014', '563 East Elliot Road, Gilbert, AZ, 85234', 'Fell
from vehicle', '09-17-2014', '09-19-2014','9770738170', 'Minor');
Insert into EMPLOYEES
         values ('0222269813', 'Lovin', 'Mc', '89021 North Jackson Street, Nashville, TN 90821', '353-243-
5081', 'T', 'Savannah James', 'Wife', '808-597-9280');

/* AFTER */
select *
   from EMPLOYEES
         where EmpID = '0222269813';

/* Basic Test #3
For a given installer/technician, we should be able to find out what certification
he has, and when they need to be renewed */

SELECT Certificate#, TradeSpecialty, DateOfRenewal
   FROM TECHNICIANS
         WHERE EmpID = '0233456789';

/* Basic Test #4
Deleting an installer/technician and all of their certifications.*/
/* Inserting on delete cascade constraints to various attributes to enable the deletion
of an employee easily and smoothly from the database */
ALTER TABLE TECHNICIANS
   DROP CONSTRAINT [FK__TECHNICIA__EmpID__671F4F74];

ALTER TABLE TECHNICIANS
   ADD FOREIGN KEY (EmpID)
   REFERENCES EMPLOYEES(EmpID)
   ON DELETE CASCADE;

ALTER TABLE WORK_ORDERS
   DROP CONSTRAINT [FK__WORK_ORDE__TechI__2645B050];

ALTER TABLE WORK_ORDERS
   ADD FOREIGN KEY(TechID)
```

```
   REFERENCES TECHNICIANS(EmpID)
   ON DELETE CASCADE;

ALTER TABLE PARTS_USED
   DROP CONSTRAINT [FK__PARTS_USE__WorkO__2739D489];
ALTER TABLE PARTS_USED
   ADD FOREIGN KEY(WorkOrderID)
   REFERENCES WORK_ORDERS(WorkOrderID)
   ON DELETE CASCADE;

/* Basic Test #4
BEFORE:
Deleting an installer/technician and all of their certifications */

SELECT T.EmpID AS 'Technician ID', E.EmpFName + ' ' + E.EmpLName AS 'Technician Name',
T.Certificate#
   FROM TECHNICIANS T
          JOIN EMPLOYEES E
            ON T.EmpID=E.EmpID
         WHERE T.EmpID = '8149842015';

/* Basic Test #4
AFTER:
Deleting an installer/technician and all of their certifications */

DELETE
   FROM TECHNICIANS
          WHERE EmpID = '8149842015';

SELECT T.EmpID AS 'Technician ID', E.EmpFName + ' ' + E.EmpLName AS 'Technician Name',
T.Certificate#
   FROM TECHNICIANS T
          JOIN EMPLOYEES E
            ON T.EmpID=E.EmpID
              WHERE T.EmpID = '8149842015';

/* Basic Test #5
For a given work order, displaying all of the parts and labor costs that pertain to it */


SELECT WO.WorkOrderID, PU.PartID, P.PartDescription AS 'Part Description', WO.TotalHours AS 'Hours
Worked', P.PartPrice AS 'Part Price'
```

```
    FROM WORK_ORDERS WO
            JOIN PARTS_USED PU
              ON PU.WorkOrderID=WO.WorkOrderID
            JOIN PARTS P
              ON P.PartID=PU.PartID
                WHERE WO.WorkOrderID = '0245421460';


/*Basic test #6*/
/*6. Trying to delete a Customer who has past work orders should fail.*/
delete
   from RESIDENTS
            where ResidentID = '3780896439';

/*BasicTest #7*/
/*Every employee must have an emergency contact.Adding an employee without emg contact info
should show an error.*/
Insert into EMPLOYEES
            values ('1212447095', 'Jack', 'Bosco', '12345 North 69th Avenue, Glendale, AZ 85308', '623-456-
9087', 'T', 'Donnie Bosco','Father', null);
Insert into EMPLOYEES
            values ('8190360250', 'John', 'Mayer', '18886 North 59th Avenue, Glendale, AZ 85308', '623-111-
0087', 'T', null,'Wife','900-156-1000');


/* Basic Test #8
Every user entered into the database must have a date retired after the date the credentials were
set up. Entering a user into the database with a date retired before date set up should fail. */
Insert into USERS
            values ('0123456788', '1212477095', 'Password', '2019-03-11', '2018-03-11');
Insert into USERS
            values ('6591784581', '8190360250', 'FavCerealCheerios!', '2019-03-12', '2018-03-12');
Insert into USERS
            values ('3056923968', '4808236737', 'XB2Ba8BvxLf_9S8Q', '2019-02-01', '2018-03-09');


/* Basic Test #9
Every part used in a work order must be some part which already exists in the database.
Using a part which does not exist in a work order will fail. */
Insert into PARTS_USED
            values ('0245421460', '0866242800');
Insert into PARTS_USED
```

```
        values ('7799342414', '0866240420');
Insert into PARTS_USED
        values ('5605315456', '0547436819');

/* Basic Test #10
Attemping to delete a property from PROPERTY table and a supplier from SUPPLIERS table.
Deleting a property from PROPERTY will fail as ASSETS and WORK_ORDERS tables require PropertyID.
Deleting a supplier from SUPPLIERS will fail as ASSETS_SUPPLIER and PARTS_PURCHASED tables require
SupplierID  */

DELETE
   FROM PROPERTY
        WHERE PropertyID = '2556275703';

DELETE
   FROM SUPPLIERS
        WHERE SupplierID = '4085239879';


/*ADVANCED QUERIES*/

/* Advanced Query #1
List the number of incidences of each accident type with more than one accident*/
SELECT AccidentType, COUNT(AccidentType) AS "Number of Accidents"
   FROM ACCIDENTS
        GROUP BY AccidentType
           HAVING COUNT(AccidentType) > 1
ORDER BY COUNT(AccidentType) DESC;

/* Advanced Query #2
List the part ID, supplier ID, supplier name, quantity in stock, and reorder point of all parts over $300*/
SELECT P.PartID, S.SupplierID, SupplierName, QuantityInStock, ReorderPoint
   FROM PARTS P JOIN PARTS_PURCHASED PP
     ON P.PartID = PP.PartID
   JOIN SUPPLIERS S
     ON PP.SupplierID = S.SupplierID
        GROUP BY P.PartID, S.SupplierID, SupplierName, QuantityInStock, ReorderPoint, PartPrice
           HAVING PartPrice > 300;

/* Advanced Query #3
List the work order ID, employee ID, and employee name of work orders with more than 10 total
hours*/
```

```
SELECT WorkOrderID, TechID, EmpFName, EmpLName
   FROM EMPLOYEES E JOIN TECHNICIANS T
    ON E.EmpID = T.EmpID
   JOIN WORK_ORDERS W
    ON T.EmpID = W.TechID
         GROUP BY WorkOrderID, TechID, EmpFName, EmpLName, TotalHours
            HAVING TotalHours > 10;

/*Advanced Query #4*/
/*List the first and last name(s) of technicians who have worked on a work order for more than 10
hours*/
Select empfname, emplname
   from employees
      where empid in
            (select empid
                  From technicians
                     where empid in
                           (select techid
                              from WORK_ORDERS
                                    where TotalHours >10));

/*Advanced Query #5*/
/*List the names of suppliers who supplied to assets located in phoenix*/
select suppliername      'Suppliers in Phoenix'
   from SUPPLIERS
         where supplierid in
            (select supplierid
                  from asset_supplier
                     where assetid in
                           (select assetid
                              from ASSETS
                                    where propertyid in
                                       (select propertyid
                                             from PROPERTY
                                                where PropertyCity = 'Phoenix')));

/*Advanced Query #6*/
/*List all the parts used for apt#444*/
select partdescription
   from PARTS
         where PartID in
            (select partid
```

```
                from PARTS_USED
                  where WorkOrderID in
                        (select workorderid
                           from WORK_ORDERS
                                 where ResidentID =
                                       (select ResidentID
                                          from residents
                                                where ResApt ='126')));

/* Advanced Test #7 - Views (Taken from the "RESULTS" section of Milestone 2)
"A listing of all users and the number of days the account was active." */
CREATE VIEW UserAccountActivity AS
  (SELECT UserID AS "User ID",
       DATEDIFF(DAY,U.DateSetUpStamp,GETDATE()) AS "Number of Days Account was/is Active"
         FROM USERS U);
SELECT *
  FROM UserAccountActivity;

/*Advanced Query #8
Total number of accidents per month for the past year by accident status. */
create view MonthsWithAccidents as
  select DATENAME(month, OccurenceDate) as "Months with Accidents",
           COUNT(DATENAME(month, OccurenceDate)) as "Number of Accidents",
           Status
         from ACCIDENTS
           where OccurenceDate > DATEADD(year,-1,GETDATE())
                 group by DATENAME(month, OccurenceDate), Status;

select *
  from MonthsWithAccidents
order by Status;

/* Advanced Test #9 - Views (Taken from the "RESULTS" section of Milestone 2)
"Residents who have had maintenance work completed in the past 6 months." */
CREATE VIEW ResidentsRecentMaintenance AS
         (SELECT R.ResidentID AS "Resident ID",
              R.ResApt AS "Resident's Apartment #",
                  R.ResName AS "Resident's Name",
                  W.WorkOrderID AS "Work Order ID",
                  W.LastMaintDate AS "Date Last Maintenance Occurred"
             FROM RESIDENTS R JOIN WORK_ORDERS W
              ON R.ResidentID = W.ResidentID
```

```
                WHERE DATEDIFF(MONTH,W.LastMaintDate,GETDATE()) <= 6);
SELECT *
        FROM ResidentsRecentMaintenance;
```

Transactions, Trigger, Procedures, Index, Visual Views

```
/* Section 1. (10) Transactions Overview:
Recall that transactions include several SQL statements coded to run in a batch execution – all statements in
the batch execute successfully (commit), or none should be executed (rollback). Your team will create 2 separate transactions using the tables/views in your project database and display the before/after screenshots
to show that the transaction was successful. */

-- TRANSACTIONS

/* 1. TRANSACTION: When a user ID is retired, an employee must be deleted or flagged as inactive. If deleted, this requires an on delete
cascade constraint with the ACCIDENTS table, which is likely undesirable for the client. If flagged as inactive, the EMPLOYEES table must
be altered to include an attribute which flags the status of the employee. The following code adds the flag attribute to the EMPLOYEES
table, making question number one possible. */
ALTER TABLE EMPLOYEES                                   -- Adds the flag attribute to the EMPLOYEES table.
        ADD DatabaseAccessStatus varchar(8);
UPDATE EMPLOYEES                                        -- Updates all employees' statuses with active access to the database.
        SET DatabaseAccessStatus = 'Active'
                WHERE EmpID IN
                        (SELECT u.EmpID
                                FROM USERS u
                                        WHERE DateRetired IS NULL);
UPDATE EMPLOYEES                                        -- Updates all employees' statuses who once had access to the database.
        SET DatabaseAccessStatus = 'Inactive'
                WHERE EmpID IN
                        (SELECT u.EmpID
                                FROM USERS u
                                        WHERE DateRetired IS NOT NULL);
UPDATE EMPLOYEES                                        -- Updates all employees' statuses who never had access to the database.
```

```
	SET DatabaseAccessStatus = 'N/A'
		WHERE EmpID IN
			(SELECT e.EmpID
				FROM EMPLOYEES e LEFT JOIN USERS u
					ON e.EmpID = u.EmpID
						WHERE u.UserID IS NULL);
SELECT * FROM EMPLOYEES;
SELECT * FROM USERS;
/* 1. When an employee has left the company, retire their userID and password so that they can no
longer
access company information and delete or flag the employee as inactive. BEFORE SCREENSHOT */
SELECT *
	FROM EMPLOYEES
		WHERE EmpID = '5425516460';
SELECT *
	FROM USERS
		WHERE EmpID = '5425516460';
/* 1. When an employee has left the company, retire their userID and password so that they can no
longer
access company information and delete or flag the employee as inactive. AFTER SCREENSHOT */
BEGIN TRANSACTION;
	UPDATE EMPLOYEES
		SET DatabaseAccessStatus = 'Inactive'
			WHERE EmpID = '5425516460';
	UPDATE USERS
		SET DateRetired = GETDATE()
			WHERE EmpID = '5425516460';
	SELECT *
		FROM EMPLOYEES
			WHERE EmpID = '5425516460';
	SELECT *
		FROM USERS
			WHERE EmpID = '5425516460';
ROLLBACK; -- If there is an error and we are not satisfied, we should choose this option.
COMMIT; -- If we are satisfied with the data, we should choose this option.


/* 2. A new supplier is added to the database. Include all parts (at least 3) that can be supplied by the
new
supplier. */

--BEFORE
```

```
select *
	from SUPPLIERS
		where SupplierID = '1234567890';

select *
	from PARTS
		where PartID = '1234567891'
		or PartID = '1234567892'
		or PartID = '1234567893';

select *
	from PARTS_PURCHASED
		where PartID = '1234567891'
		or PartID = '1234567892'
		or PartID = '1234567893';

BEGIN TRANSACTION;

insert into SUPPLIERS
	values ('1234567890', 'Daniel Jones', '11198 East Camina Plata', 'Queen Creek', 'AZ', '85142',
'480-924-7013', 'Rock Lee', 'danieljones@gmail.com', 'on-time');

insert into PARTS
	values('1234567891', 'Pumps', '1” Diaphragm Pump', 'Stainless Steel w/ Santoprene Diaphragm',
'5', NULL, '2', '2', 450.00);
insert into PARTS
	values('1234567892', 'Pumps', '1.5” Diaphragm Pump', 'Stainless Steel w/ Santoprene
Diaphragm', '5', NULL, '2', '2', 500.00);
insert into PARTS
	values('1234567893', 'Pumps', '3” Diaphragm Pump', 'Stainless Steel w/ Santoprene Diaphragm',
'5', NULL, '3', '3', 550.00);

insert into PARTS_PURCHASED
	values('1234567890', '1234567891');
insert into PARTS_PURCHASED
	values('1234567890', '1234567892');
insert into PARTS_PURCHASED
	values('1234567890','1234567893');

--AFTER
select *
	from SUPPLIERS
```

```
                where SupplierID = '1234567890';

select *
        from PARTS
                where PartID = '1234567891'
                or PartID = '1234567892'
                or PartID = '1234567893';

select *
        from PARTS_PURCHASED
                where PartID = '1234567891'
                or PartID = '1234567892'
                or PartID = '1234567893';

ROLLBACK; -- incase there is an error
COMMIT;

--TRIGGER

/* 1. (10) TRIGGER: When a new work order is created (inserted), automatically assign the first available technician to do the job; assign
a tech with the appropriate tech specialty (if the task is plumbing, don't assign an electrician). */
/* Before we create the trigger, we must create a way to flag which technicians are available and which are inavailable from working
on a work order. To do this, we must create a flag attribute as we did in the first transaction question. */
ALTER TABLE TECHNICIANS                                         -- Adds the flag attribute to the TECHNICIANS table
        ADD AvailabilityStatus varchar(11);
UPDATE TECHNICIANS                              -- Since all work orders are finished in the Work Orders Table,
        SET AvailabilityStatus = 'Available'        -- all technicians are available.
                WHERE EmpID IN
                        (SELECT TechID
                                FROM WORK_ORDERS
                                        WHERE EndDate IS NOT NULL);

UPDATE TECHNICIANS                                              -- Since all work orders are finished in the Work Orders Table,
        SET AvailabilityStatus = 'Unavailable'                  -- no technicians should be unavailable.
                WHERE EmpID IN
                        (SELECT TechID
```

```
                                FROM WORK_ORDERS
                                        WHERE EndDate IS NOT NULL);

SELECT * FROM TECHNICIANS;

/* Next, we must insert a value to into Tasks which shows the correct Task Type. If a correct task type is
provided, it can be used to
match an available technician. */
INSERT INTO TASKS
        VALUES('1039259302','HVAC','Repair Damaged Air Filters', 'Wear and Tear');
SELECT * FROM TASKS;

/* Creating the trigger (with successful execution message) */
CREATE TRIGGER TechniciansWOInsert
ON WORK_ORDERS
AFTER INSERT
AS
        UPDATE WORK_ORDERS
                SET TechID =
                        (SELECT TOP 1 EmpID
                                        FROM TECHNICIANS
                                                WHERE TradeSpecialty IN
                                                        (SELECT TaskType
                                                                FROM TASKS
                                                                        WHERE TaskID IN
                                                                                (SELECT TaskID
                                                                                        FROM inserted)
                                                                        AND AvailabilityStatus = 'Available'))
                        WHERE WorkOrderID IN
                                (SELECT WorkOrderID
                                        FROM inserted);
        UPDATE TECHNICIANS
                SET AvailabilityStatus = 'Unavailable'
                        WHERE EmpID IN
                                (SELECT TechID
                                        FROM WORK_ORDERS
                                                WHERE WorkOrderID IN
                                                        (SELECT WorkOrderID
                                                                FROM inserted));

/* 1. (10) TRIGGER: When a new work order is created (inserted), automatically assign the first available
technician to do the job; assign
```

```
a tech with the appropriate tech specialty (if the task is plumbing, don't assign an electrician). */
/* Before screenshot */
SELECT *
	FROM WORK_ORDERS
		WHERE WorkOrderID = '8437549859';
SELECT * FROM TECHNICIANS;

/* 1. (10) TRIGGER: When a new work order is created (inserted), automatically assign the first available
technician to do the job;
assign a tech with the appropriate tech specialty (if the task is plumbing, don't assign an electrician). */
/* After screenshot */
INSERT INTO WORK_ORDERS
	VALUES ('8437549859', '6393256990', '9534546401', '1039259302', NULL, '3663027200', '2020-
04-28', '2020-04-28',
	'2020-05-05', NULL, NULL, '0', 23.38, 'Damaged Filter', 'Completed');
SELECT * FROM WORK_ORDERS
	WHERE WorkOrderID = '8437549859';
SELECT * FROM TECHNICIANS;

--PROCEDURE

/* (10) PROCEDURE:  When a work order status changes to ‘delayed’, add a record to the
DelayedWOLog and record WO#, StartDate, tech working on the order (ID),
status change date (today’s date), root cause for the WO, and reason for the delay.

PROCEDURE: Creating table DelayedWOLog to log when WorkOrder Status is changed to delayed  */

CREATE TABLE DelayedWOLog (
WO#			char(10),
StartDate		date,
TechID			char(10),
StatusChangeDate	datetime		DEFAULT GETDATE(),
RootCause		varchar(1000),
DelayReason		varchar(1000));

SELECT *
	FROM DelayedWOLog;

/* (10) PROCEDURE:  When a work order status changes to ‘delayed’, add a record to the
DelayedWOLog and record WO#, StartDate, tech working on the order (ID),
status change date (today’s date), root cause for the WO, and reason for the delay.
```

```
PROCEDURE: Creating procedure AddDelayedWO to update DelayedWOLog  */
 CREATE PROCEDURE AddDelayedWO
                @WorkOrderID char(10),
                @DelayReason  varchar(1000)
        AS
                INSERT INTO DelayedWOLog(WO#, StartDate, TechID, StatusChangeDate, RootCause,
DelayReason)
                        VALUES (@WorkOrderID,
                                (SELECT StartDate
                                        FROM WORK_ORDERS
                                                WHERE WorkOrderID = @WorkOrderID
                                                  AND (WorkOrderStatus = 'Delayed' OR
WorkOrderStatus = 'delayed')),
                                (SELECT TechID
                                        FROM WORK_ORDERS
                                                WHERE WorkOrderID = @WorkOrderID
                                                  AND (WorkOrderStatus = 'Delayed' OR
WorkOrderStatus = 'delayed')),
                                GETDATE(),
                                (SELECT RootCause
                                        FROM TASKS
                                                WHERE TaskID IN
                                                        (SELECT TaskID
                                                                FROM WORK_ORDERS
                                                                        WHERE WorkOrderID =
@WorkOrderID
                                                                  AND (WorkOrderStatus = 'Delayed' OR
WorkOrderStatus = 'delayed'))),
                                @DelayReason);

EXEC AddDelayedWO 2393881879, 'Improper tools';
EXEC AddDelayedWO 9375544818, 'Missing part';

SELECT *
        FROM DelayedWOLog;

--INDEXES AND OPTIMIZATION

/*1. (2) Identify the attributes (at least 2) that should have secondary indexes defined (cluster index is
already created) to speed up the query you will define in #3. Take a snapshot of the tables you will use (
select * from each table and show the attributes and a few rows of data in each one). */
SET STATISTICS TIME ON;
```

```
SELECT *
	FROM WORK_ORDERS;
SELECT *
	FROM RESIDENTS;

/* 2. (2) Write the SQL commands to create the secondary indexes for the 2 attributes identified in #1.
Give reasons for each attribute selected (use comments). Take a snapshot of the SQL code for both
indexes. */
--StartDate is a constraint in the Where clause, and using an index could reduce sorting and processing
time.
CREATE INDEX StartDateIDX ON WORK_ORDERS(StartDate);

--ResName is a constraint in the ORDER BY clause, indexing ResName could reduce sorting and
processing time as well.
CREATE INDEX ResNameIDX ON RESIDENTS(ResName);


/*3. (2) Create an advanced join query and execute (use any multiple tables you wish). Show the query,
the
data results, the time to complete, and the query plan (take snapshots of each). The idea here is to
create a query that will tax the computer resources – force the DBMS to use its resources. */

SELECT r.ResidentID AS "ID", r.ResApt, r.ResName, r.ResMobilePhone, r.ResEmail --RESIDENT
	FROM RESIDENTS r
		JOIN WORK_ORDERS w
			ON r.ResidentID = w.ResidentID
		JOIN PROPERTY p
			ON w.PropertyID = p.PropertyID
	WHERE YEAR(StartDate) = 2019
	AND YEAR(EndDate) = 2019
	AND p.PropertyState IN ('AZ')
	AND ResApt > 100
ORDER BY ResName;

/*4. (2) Optimize & execute the query in #3. Show the query, the data results, the time to complete, and
the query plan (take snapshots of each). */
SELECT r.ResidentID, r.ResApt, r.ResName, r.ResMobilePhone, r.ResEmail --RESIDENT
	FROM RESIDENTS r
		WHERE ResApt > 100
		AND r.ResidentID IN
			(SELECT w.ResidentID
				FROM WORK_ORDERS w
```

```
                                        WHERE YEAR(StartDate) = 2019
                                        AND YEAR(EndDate)= 2019
                                        AND w.PropertyID IN
                                                (SELECT p.PropertyID
                                                        FROM PROPERTY p
                                                                WHERE p.PropertyState IN ('AZ')))
ORDER BY ResName;


--DATA VISUALIZATION VIEWS

/* 1. (4) A screenshot from SQL Server Management Studio of the view created by each team member
and
the data in the view. Show the Create View statement and contents of the view (.pdf file).
Create a view that produces a fact table from your team's project database

Mat Brewer
STORY: Using Treemap to categorize which Supplier sold the most of each category.*/
CREATE VIEW MatsVisualization AS
SELECT p.PartID, PartCategory, QuantityInStock, ReorderPoint, ReorderQuantity, PartPrice,
                s.SupplierID, SupplierName, SupplierState, DeliveryRating
        FROM PARTS p
                JOIN PARTS_PURCHASED pp
                        ON p.PartID = pp.PartID
                JOIN SUPPLIERS s
                        ON s.SupplierID = pp.SupplierID;
SELECT *
        FROM MatsVisualization;


/* 1. (4) A screenshot from SQL Server Management Studio of the view created by each team member
and
the data in the view. Show the Create View statement and contents of the view (.pdf file).
Create a view that produces a fact table from your team's project database

Varun Shourie, CIS365, Section 19668
STORY: Using Treemap to Lists the total hours spent on work orders by residents and properties.*/

CREATE VIEW VarunPropResWO AS
        SELECT R.ResidentID, R.ResApt, R.ResName, R.ResPhone, R.ResMobilePhone, R.ResEmail, --
RESIDENTS table attributes.
```

```
                WO.WorkOrderID, WO.TaskID, WO.CreatorID, WO.DateCreated, -- WORK ORDER table
attributes
                WO.DesiredWorkDate, WO.StartDate, WO.EndDate, WO.LastMaintDate,
WO.Emergency, WO.TotalHours,
                WO.WorkOrderNotes, WO.WorkOrderStatus,
                P.PropertyID, P.PropertyName, P.PropertyAddress, P.PropertyCity, P.PropertyState, --
PROPERTY table attributes.
                P.PropertyZIP, P.PropertyManager, P.PropertyPhone, P.PropertyAccountNo,
P.PropertyTech
            FROM RESIDENTS R JOIN WORK_ORDERS WO
                        ON R.ResidentID = WO.ResidentID
                JOIN PROPERTY P
                        ON WO.PropertyID = P.PropertyID;
SELECT *
        FROM VarunPropResWO;

/* 1. (4) A screenshot from SQL Server Management Studio of the view created by each team member
and
the data in the view. Show the Create View statement and contents of the view (.pdf file).
Create a view that produces a fact table from your team's project database

Isis Sanchez - Creating view TechnicianAccidents and join data in Employees, Accidents, and Technicians
Story: Employee Accidents Among Techician Specialties by Severity */

CREATE VIEW TechnicianAccidents AS
        SELECT E.EmpFName, E.EmpLName,
                A.EmpID, A.AccID, A.OccurenceDate, A.AccidentAddress, A.AccidentType,
A.WorkSuspDate, A.ResumeDate, A.Status,
                T.TradeSpecialty
          FROM EMPLOYEES E JOIN ACCIDENTS A
                ON E.EmpID = A.EmpID
           JOIN TECHNICIANS T
                ON A.EmpID = T.EmpID;

SELECT *
        FROM TechnicianAccidents;
/* M5 Revisions - Connect the Employees table to the Users table. - Varun Shourie */

ALTER TABLE USERS
ADD FOREIGN KEY(EmpID)
REFERENCES EMPLOYEES(EmpID);
```

```
/*NTV M-5*/
/* 1. (4) A screenshot from SQL Server Management Studio of the view created by each team member
and
the data in the view. Show the Create View statement and contents of the view (.pdf file).
Create a view that produces a fact table from your team's project database */

Create View propertiesbylocation
as
	select propertyid,propertycity
		from PROPERTY;

select *
	from propertiesbylocation
order by PropertyCity;

/* 1. (4) A screenshot from SQL Server Management Studio of the view created by each team member
and
the data in the view. Show the Create View statement and contents of the view (.pdf file).
Create a view that produces a fact table from your team's project database

Janae Lewis
STORY: Lists the Work Orders by the number of parts used and property.*/

CREATE VIEW WONumberOfPartsUsed AS
	SELECT W.WorkOrderID,
		PR.PropertyID, PR.PropertyName,
		COUNT(PartID) AS "Number of Parts Used"
	 FROM PARTS_USED P JOIN WORK_ORDERS W
		ON P.WorkOrderID = W.WorkOrderID
	 JOIN PROPERTY PR
		ON W.PropertyID = PR.PropertyID
		GROUP BY W.WorkOrderID, PR.PropertyID, PR.PropertyName;

SELECT *
	FROM WONumberOfParts
/* M5 Revisions - Connect the Employees table to the Users table. - Varun Shourie */
ALTER TABLE USERS
ADD FOREIGN KEY(EmpID)
REFERENCES EMPLOYEES(EmpID);
```

```
/*Selecting all tables to show the data within*/
--Select from ACCIDENTS table
SELECT *
	FROM ACCIDENTS

--Select from ASSET_SUPPLIER table
SELECT *
	FROM  ASSET_SUPPLIER

--Select from ASSETS table
SELECT *
	FROM ASSETS

--Select from DelayedWOLog table
SELECT *
	FROM DelayedWOLog

--Select from EMPLOYEES table
SELECT *
	FROM EMPLOYEES

--Select from OFFICE_STAFF table
SELECT *
	FROM OFFICE_STAFF

--Select from PARTS table
SELECT *
	FROM PARTS

--Select from PARTS_PURCHASED table
SELECT *
	FROM PARTS_PURCHASED

--Select from PARTS_USED table
SELECT *
	FROM PARTS_USED

--Select from PROPERTY table
SELECT *
	FROM PROPERTY

--Select from RESIDENTS table
```

```
SELECT *
	FROM RESIDENTS

--Select from SUPPLIERS table
SELECT *
	FROM SUPPLIERS

--Select from TASKS table
SELECT *
	FROM TASKS

--Select from TECHNICIANS table
SELECT *
	FROM TECHNICIANS

--Select from USERS table
SELECT *
	FROM USERS

--Select from WORK_ORDERS table
SELECT *
	FROM WORK_ORDERS
```

APPENDIX B:

Team Resumes, Journal Entries, Team Histogram

Resumes

Isis Sanchez

Gilbert, AZ
602-400-9259
isistsanchez@gmail.com

Skills

- Bilingual: Spanish (Native)
- Knowledge of Python, SQL, SPSS
- Microsoft Office Suite – Excel, Access
- Customer Service/Engagement
- **Awards**: W.P. Carey School of Business Dean's List, Academic Achievement Scholar

Experience & Community Involvement

DECEMBER 2018 – PRESENT

Sales Associate (Marshalls) / Gilbert, AZ

Effectively servicing customers while remaining flexible to ongoing needs of the business.

- Trained staff during register enhancements while maintaining work tasks
- Fulfilled collaborative/individual projects
- Increased sales through customer credit card enrollments

SEPTEMBER 2019 – PRESENT

Department of Information Systems Club (DISC) / Tempe, AZ

Current member

- Attended weekly meetings to network and gain insight from industry professionals
- Volunteer work including Feed My Starving Children and Community Instruction Program

Education

MAY 2021

Bachelor of Science - Business Data Analytics

W.P. Carey School of Business — GPA: 3.6
Arizona State University, Tempe, AZ

MAY 2021

Bachelor of Arts - Business Human Resource

W.P. Carey School of Business — GPA: 3.6
Arizona State University, Tempe, AZ

References

[Available upon Request]

Varun Shourie vshourie@asu.edu | 623-308-4818| Canadian citizen in the U.S. (OCI)
Glendale, AZ https://www.linkedin.com/in/varunshourie/

EDUCATION

Arizona State University, Tempe, AZ *May 2021*
W. P. Carey School of Business | Barrett, the Honors College
Bachelor of Science, Computer Information Systems
CGPA: 4.0/4.0

- Honors/Awards: New American University Scholarship, Leaders' Academy Scholar, Dean's List since Fall 2018.
- Relevant courses: CIS340 (Java), CIS365 (Database Systems), CIS401 (Cyber Risk Management), CIS236 (Honors Intro to Information Systems), SCM300 (Supply Chain Management), MGT300 (Organizational Management & Leadership), ACC231 (Financial Accounting), ACC241 (Managerial Accounting).

PROFESSIONAL EXPERIENCE

HP Law, Glendale, AZ *May 2019 – August 2019*
Legal Intern/Clerk

- Finished work projects such as drafting documents such as Revocable Living Trusts, Last Will and Testaments, Power of Attorney documents, Deeds, Probate Court pleadings, and business formation and operating agreements for a total of approximately 25 clients, reducing the average cycle time of cases by about 2-3 days.
- Managed the upkeep and workflow of approximately 50 clients by interfacing with technologies like CRM systems in a team with the paralegal and attorney, allowing for 100% required documentation of all activities.

VOLUNTEER EXPERIENCE

Glendale Foothills Library, Glendale, AZ *May 2014 – Present*

- Built technology literacy through guiding and communicating with senior citizens in weekly computer classes (e.g. Word, Excel, web browsing), helping approximately 5-10 patrons per class for ~20 classes so far.
- Managed flow of books by helping shelve on average 5 carts of books per session and analyzing patron data for trends, reducing employee time spent on circulation by approximately 1 hour every single day volunteered.
- Coordinated the creation of biannual book sales by keeping track of book inventories, ordering books from other library branches, and conducting heavy manual labor, reducing set up time from 5 days to approximately 2 days.

St. Mary's Food Bank and **Desert Mission Food Bank**, Phoenix, AZ *May 2016 – Present*

- Oversaw the creation of food carts and shipments distributed to the needy by assembling packages and tracking the number of packages provided to food bank customers, serving approximately 40+ customers per day served.

ACTIVITIES AND INVOLVEMENT

On Campus-Clubs: *August 2018 – Present*

- **IT Committee Member** at **Department of Information Systems Club:** Assisted the Vice President of IT of DISC in duties by helping with small coding projects, completing ad hoc projects for other committees, and managing the club website as necessary, helping the club expand membership by approximately 10 more students.
- Member at **Software Developers' Association:** prepared for technical career in IT by participating in mock technical interviews and events with recruiters from big tech firms (e.g. Microsoft, Amazon, Google, etc.), building my network by 15 recruiters.

SKILLS AND HOBBIES

Microsoft Office Suite: Excel, Word, and PowerPoint.

Technical Knowledge/Skills: SQL, Java, database concepts (Spring 2020), SDLC basics (Spring 2020), and cybersecurity basics (Spring 2020).

Personal Interests: Running, Hiking, Basketball, Travel in Northern AZ/CA, India.

NAYAN TEZ

977 E Apache Blvd, Tempe AZ
E -mail: **9tez99@gmail.com**/9tez@asu.edu Phone**: 480-740-0505**

Objective:

Currently looking for an internship in the supply chain management industry to get equipped with knowledge & global exposure, while finishing up my undergraduate study at W. P. Carey School of Business.

Experience:

- **Supply Chain Intern**
  Synergies Casting Limited, Vizag SEZ, India (May2019-Aug2019)
  - Worked closely with the supply chain department at a major automotive alloy wheels OEM in India understanding the SCM practices followed by Ford, GM and other major auto manufacturers.
  - Worked with Ford's auto plants in India and updated their portal with delivery information and documents such as ASN, dispatch schedules etc. and maintain JIT delivery system for efficiency at ford's plant.
  - Made a project report detailing the benefits of using in house packaging options and manage their reverse logistics for recycling, instead of using CHEP (Brambles.Ltd), the current 3rd party company currently handling our packaging.
- **Yoga Instructor**
  Yoga Village, Visakhapatnam, India (May2017-May2019)
  - Worked part-time at the yoga village while assisting the guru with children.
  - Made instructional and educational videos about yoga and its benefits for his website.
- **Front Desk**
  Dolphin Diagnostics Services, India (May2018 -Aug2018)
  - Worked at the front desk reception handling responsibilities such as Customer service, Data entry, Reports dispatch logistics, event organization.
  - Currently helping as a consultant to choose database software and hire firms for database creation in order keep track of referrals and daily volume.

Education:

GPA/Grade

- **Supply Chain Management (BS)** **3.46/4.0**
  W. P. Carey School of Business, Arizona State University, Tempe. (2017-2021)
- **Business Data Analytics (BS)** **3.46/4.0**
  W. P. Carey School of Business, Arizona State University, Tempe. (2017-2021)
- **High- School (CBSE,BiPC)** **9.0/10**
  Oakridge International School, Vizag, India.(2012-2016)

Skills:

- **M.S Office Suite**
  - Proficient in basic and complex Excel functions for descriptive, predictive and prescriptive analytics.
  - Data visualizations like pivot tables, histograms etc.
- **Tableau**
  - Familiar with working in the software and exporting live or offline excel documents.
  - Capable of creating complex dashboard with large data to make meaningful visualizations.
- **SQL**
  - Skilled at writing complex queries in MS SQL.
  - Familiar with EERD diagrams and database creation.

Personal details:

- **Nationality**: Indian
- **Languages Known**: English, Telugu, Hindi, Sanskrit and familiar with French.
- **Hobbies**: Squash, Swimming, Automotive Racing, Gourmet Cooking.

Mathew Brewer

matbrewerwork@gmail.com | (480)294-0317 | Chandler, Arizona | LinkedIn |

SUMMARY

Adaptable Computer Information Systems major currently attending Arizona State University, with 2+ years of relevant coursework. Seeking an opportunity to leverage and grow knowledge of programming, problem solving, and database design skills to successfully fill the Internship role at your company. Known as being analytical and efficient, I can be a viable asset to help your organization achieve its goals.

EDUCATION

Arizona State University — **Aug 17 - May 21**
BS, Computer Information Systems — *Tempe, AZ*

- 3.75/4.0 GPA; 4x Deans List (2017-2019); Provosts Scholarship
- **Relevant Coursework**:
  - CIS 340: Business Systems Development I (Java)
  - CIS 345: Business System Development II (Python)
  - CIS 365: Database Systems

SKILLS & INTERESTS

Languages:

- Java (Intermediate level)
- Python (Intermediate level)
- SQL

Interests: Japanese Culture and Language; New technology; Gaming; Meeting new people; Playing volleyball; Weightlifting.

Soft Skills:

- Adapting to and applying newly learned skills
- Communication within a team
- Problem solving
- Contract negotiations
- Account management
- Customer service

WORK EXPERIENCE

Big 5 Sporting Goods Corp. — **Jun. 2018 – Present**
Sales Associate — *Chandler, AZ*

- Maintain knowledge of current promotions, sales, payment policies, and security application.
- Compute sale prices, total purchases and receive and process cash or credit payment.
- Greet Customers and ascertain the wants and needs of each customer.

Anytime Fitness — **Sep. 2018 – Mar. 19**
Part-time Manager — *Phoenix, AZ*

- Oversight of the entire club membership base including operational and financial responsibilities, back-end reporting and management of member needs.
- New member acquisition through direct 1-1 sales presentations.
- Oversight of the club's maintenance and upkeep.

Revelation Real Estate — **Sep. 17– Jan. 19**
Real Estate Agent — *Chandler, AZ*

- Comparative analysis to determine the competitive market price of a property.
- Show properties to clients, discuss and negotiate conditions of purchase or sale, and draw up real estate contracts.

MISCELLANEOUS

- **Certifications/Licenses:** Real Estate License in Arizona

Janae Lewis
Scottsdale, AZ | 480-239-7824 | jalewi26@asu.edu

EDUCATION

- **Arizona State University, Tempe, AZ**
  B. S., Computer Information Systems
  Expected graduation date: May 2021
- Chinese Language Flagship scholar
- Barrett, The Honors College student
- Awards: Provost's Scholarship, Dean's List, Flagship Summer Study Abroad Scholarship

PROFESSIONAL EXPERIENCE

Market Research Intern—Biopharmaceutical Division **May 2019-October 2019**
Govig & Associates, Scottsdale, AZ

- Composed research projects on industry trends in the biopharmaceutical sector with an emphasis on new and emerging companies that specialize in oncology, orphan drugs, and rare diseases
- Managed the company SQL database to ensure that data was relevant and reliable for the organization
- Designed Excel spreadsheets to organize information on the biopharmaceutical team's network of industry professionals
- Performed candidate research for specific roles using recruiter software
- Trained incoming project coordinators on utilizing the database and on market research methods

Customer Service Specialist—Student Services **October 2018-April 2019**
University Technology Office, Arizona State University, Tempe, AZ

- Responded to customer inquiries about university programs and resources via phone and chat software
- Utilized ServiceNow knowledge base to find accurate information, thus enhancing the customer service experience
- Documented cases using ServiceNow and Salesforce CRM applications

CAMPUS/COMMUNITY INVOLVEMENT

Tour Guide **January 2018-Present**
Devils' Advocates, Tempe, AZ

- Inform prospective students and families of campus life and opportunities at Arizona State University through personalized campus tours
- Conduct special tours for high school and community college groups, non-prospective visitors, and alumni
- Act as a member of the training committee and assist with planning training for new guides

Volunteer **June 2017-Present**
PetSmart Charities, Scottsdale, AZ

- Assist with shelter maintenance and animal care at a PetSmart Charities cat shelter

ADDITIONAL SKILLS

- Programming languages: Python, HTML/CSS, Java, JavaScript
- Frameworks: Bootstrap
- Proficiency in MS Excel, MS SQL Server, Tableau, and MS Access
- Spoken languages: English (native), Mandarin Chinese (advanced), Brazilian Portuguese (intermediate)

CERTIFICATIONS

- Responsive Web Design, freeCodeCamp

Personal Journal Entries

Team Member Name: Mat Brewer		**Initials: MB**
Date of Work Performed	**Description of Work Performed**	**Time Spent on Work Performed (HRS)**
1/25/20	Remade my own resume	3
1/27/20	Member intro paragraph for M1	0.25
1/27/20	Attended Team Meeting (did Membership Agreement, Team Profile, Member Roles)	1.15
1/31/20	Proofread M1 for errors before submitting	.1
2/10/20	Chaired for Milestone 2 meeting	1
2/10/20	Added basic entity and attributes to EERD with relationships	2
2/14/20	Worked on the data dictionary by adding data types and lengths of data. Also improved description of some attributes.	2
2/15/20	Proofread Milestone 2 before submission	0.25
3/12/20	Created Metadata for ASSETS, ASSET_SUPPLIER, SUPPLIER, USERS, PARTS_PURCHASED, PROPERTY, and PARTS_USED tables.	1
3/13/20	Created Python scripts to automate certain sample data.	2
3/15/20	Created sample data for SUPPLIERS and RESIDENTS tables.	3
3/18/20	Created sample data for TECHNICIANS table	1
3/22/20	Queries to test Entity Integrity	1
3/22/20	Fix last minute bugs in database	1
3/22/20	Proofread Milestone 3 before submission	0.25
4/2/20	Did Advanced Queries 7-9 with Varun	1

Team Member Name: Mat Brewer		**Initials: MB**
Date of Work Performed	**Description of Work Performed**	**Time Spent on Work Performed (HRS)**
4/4/20	Provided Queries to show the before and after effects of basic tests 1 & 2	.5
4/5/20	Compiled and edited txt file	1
4/5/20	Proofread M4 document	.25
4/9/20	Created Data Visualization View and Treemap	1
4/14/20	Created Index Query and Index	1
4/16/20	Changed Index query to a nested query to test the changes in Query Plan	.5
4/16/20	Created the Transaction for inserting a new supplier	1
4/24/20	Create the analytics dashboard in Tableau	.5
4/26/20	Made changes to screenshots from ICs to make them conform with M5 standards	.25
4/26/20	Compiled the txt file	.25
4/26/20	Proofread M5	.25
4/27/20	Finished Franklins Requirements	.5
4/27/20	Added Query screenshot section of M7	.25
4/28/20	Created slides/script for my part of M6	1.5
4/29/20	Added sample screenshot of every table in the database	.50
5/1/20	Created Appendix A/Txt File	.5
5/1/20	Populated Various sections with content from past milestones.	1.1
5/3/2020	Proofread M7 prior to submission	0.5
	Total Hours for Mat:	**31.5**

Team Member Name: Varun Shourie		Initials: VS
Date of Work Performed	**Description of Work Performed**	**Time Spent on Work Performed (HRS)**
1/25/20	Updated resume by adding skills, removing old positions, and rewording descriptions	0.50
1/26/20	Created Team Logo for Milestone 1	0.50
1/27/20	Intro paragraph written for Nayan Tez	0.30
1/27/20	Chaired Meeting (did Membership Agreement, Team Profile, Team Roles, and Meeting Agenda)	1.25
1/31/20	Proofed and fixed mistakes (e.g. formatting, improper responses) in Milestone 1 prior to submission	1.25
2/9/20	Prepared for Team Meeting 02 by annotating milestone specs and .xlsx file	0.30
2/10/20	Started working on Data Dictionary and EERD in Team Meeting 02 with members	1.25
2/11/20	Clarified doubts about business rules with Dr. Huang in office meeting	0.75
2/12/20	Proofed the EERD and added missing attributes from sample Excel data.	1.20
2/14/20	Added attributes and completed definitions in the Data Dictionary	1.00
2/15/20	Added attributes/definitions and finished the Data Dictionary	4.50
2/16/20	Proofed M2 before submitting on Canvas	0.25
2/16/20	Fixed resource histogram made by VNT	0.25
3/5/20	Fixed multivalued attributes on EERD	0.25
3/9/20	Completed relational schema	1.50
3/11/20	Did “insert into” scripts for USERS & EMPLOYEES tables	1.50

Team Member Name: Varun Shourie		**Initials: VS**
Date of Work Performed	**Description of Work Performed**	**Time Spent on Work Performed (HRS)**
3/12/20	Did “insert into” scripts for OFFICE_STAFF	0.50
3/13/20	Did “insert into” scripts for PARTS	0.75
3/15/20	Debugged metadata	0.40
3/17/20	Did “insert into” scripts for PARTS_USED	0.50
3/19/20	Added further constraints to metadata in database scripts	0.40
3/21/20	Executed all CREATE/INSERT scripts on personal database and debugged them	2.25
3/21/20	Wrote scripts to test entity + referential integrity for the database	2.00
3/22/20	Proofread M3 prior to submission	0.30
3/22/20	Updated hours on histogram	0.05
3/31/20	Wrote basic queries 8 + 9	0.50
4/1/20	Debugged Queries 4 + 5 with Isis (added on delete cascade to metadata)	0.40
4/2/20	Finished views with Mat (advanced queries 8 + 9)	0.65
4/3/20	Commented/gave feedback on query contents to all group members	0.50
4/5/20	Proofread M4 prior to submission, added team member hours to histogram.	0.15
4/9/20	Created Tree Map visualization	0.62
4/16/20	Completed transaction #1	1.00
4/18/20	Completed a draft/skeleton of the logic for creating the procedure for Isis	0.53
4/19/20	Attended meeting with Dr. Nina to clarify M5 instructions.	0.40

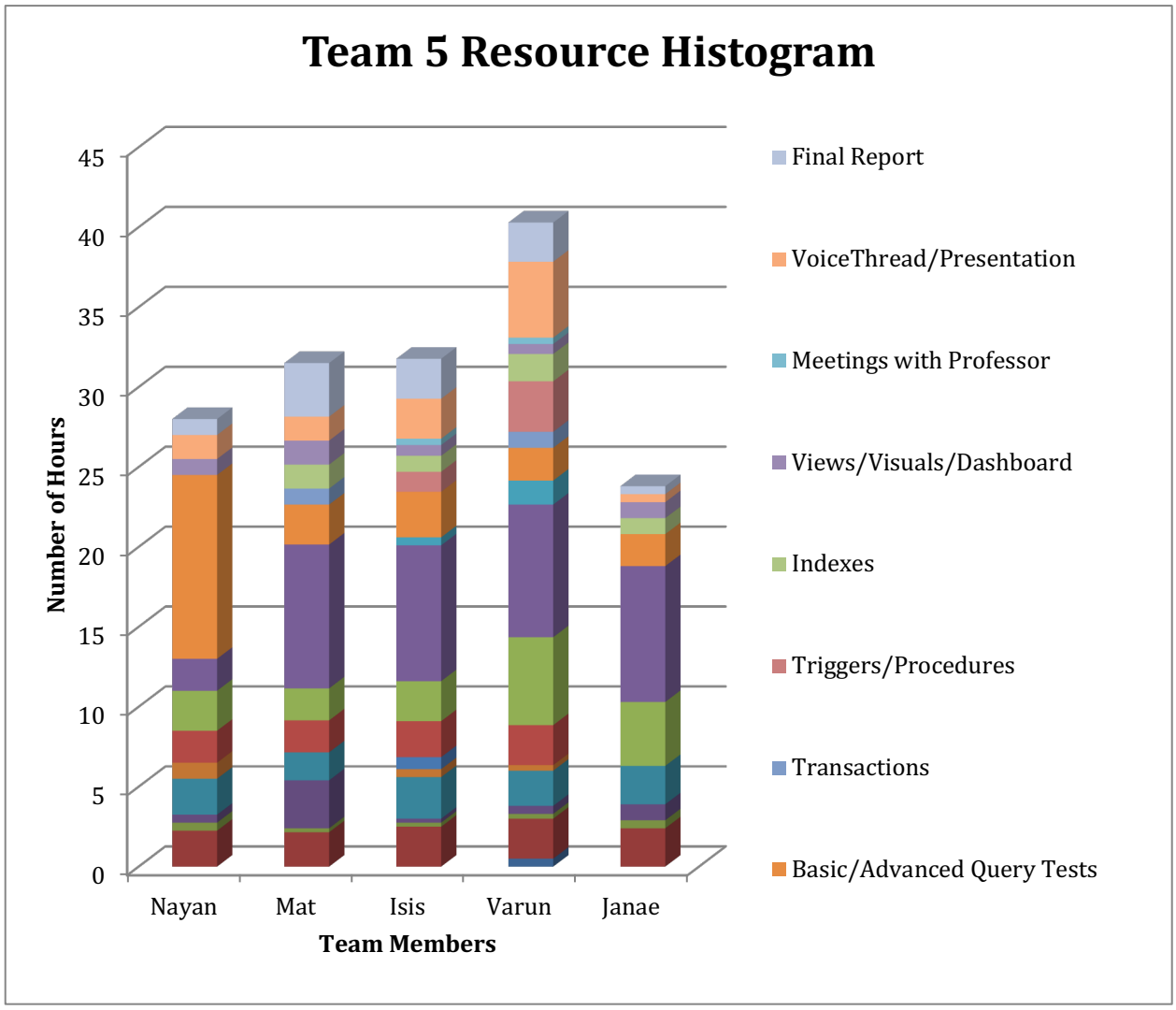
Team Member Name: Varun Shourie		**Initials: VS**
Date of Work Performed	**Description of Work Performed**	**Time Spent on Work Performed (HRS)**
4/23/20	Wrote an additional advanced join query on top of the one from the exercise and optimized it (not used in the milestone)	1.00
4/25/20	Debugged trigger #1, repaired DB structure from errors caused by another teammate in the database, and completed trigger #1.	2.63
4/25/20	Wrote the explanation for number five of Mat's query optimization.	0.71
4/26/20	Proofread M5 prior to submission.	0.30
4/29/20	Completed my portion of the presentation on Google Slides.	2.50
4/30/20	Wrapped up the Google Slides presentation (including others' parts).	0.82
4/30/20	Recorded approximately five minutes of presentation time on VoiceThread	0.83
5/1/20	Cut down presentation time on VoiceThread with edits.	0.60
5/3/2020	Formatted Milestone 7	1.95
5/3/2020	Proofread M7 prior to submission	0.50
	Total Hours for Varun:	**40.29**

Team Member Name: Janae Lewis		**Initials: JL**
Date of Work Performed	**Description of Work Performed**	**Time Spent on Work Performed (HRS)**
1/19/20	Posted and edited resume	1
1/19/20	Wrote Varun's personal biography	0.5
1/27/20	Attended Meeting (did Membership Agreement, Team Profile, Team Roles)	1.15
2/1/20	Proofread Milestone 1 for any errors before submitting	0.5
2/10/20	Team Meeting	1.25
2/11/20	Worked on data dictionary	2
2/15/20	Edited data dictionary	2
2/16/20	Proofread Milestone 2 before submitting	0.2
3/10/20	Created tables	.5
3/11/20	Created tables	2
3/13/20	Created sample data for SQL tables	2.5
3/18/20	Created sample data for SQL tables	2.5
3/21/20	Composed join queries	1
3/22/20	Proofread M3 before submission	1
4/2/2020	Create advanced queries	2
4/5/2020	Proofread M4 before submission	0.2
4/9/2020	Completed data visualization	1
4/14/2020	Completed index (not included in final document)	1
4/26/2020	Proofread M5	0.5
5/1/2020	Recorded VoiceThread	0.5
5/2/2020	Proofread M7	0.5
	Total hours for Janae:	**23.8**

Team Member Name: Isis Sanchez		**Initials: IS**
Date of Work Performed	**Description of Work Performed**	**Time Spent on Work Performed (HRS)**
1/21/20	Set up Google Docs and shared resources	0.5
1/26/20	Updated Resume with new skills/positions	0.25
1/27/20	Attended Team Meeting (did Membership Agreement, Profile, Member Roles)	1.25
1/27/20	Finished Code of Ethics (editing)	0.25
1/27/20	Finished Member Profile	0.25
1/31/20	Proofread Milestone 1 for any errors before submitting.	0.50
2/9/20	Set up Team Google Drive	0.25
2/9/20	Read/Highlight Key attributes, entities, and relationships in client description	1.5
2/10/20	Team Meeting (Began collaboration on ERD)	1.25
2/11/20	Discussed business rules with Dr. Huang in office meeting and edited ERD	0.75
2/12/20	Proofread EERD	0.25
2/15/20	Proofread and edited EERD	0.5
2/15/20	Proofread and added missing attributes, entities to Data Dictionary	2.5
2/15/20	Proofread M2 before submission	0.25
3/8/2020	Made revisions to EERD/ Data Dictionary	0.5
3/11/2020	Restored Data Dictionary, created tables, and edited tables	1.5
3/12/2020	Created sample data for ACCIDENTS table	1
3/13/2020	Created sample data for TASKS and WORK_ORDERS tables	2
3/16/2020	Created PropertyIDs for sample data for PROPERTY table	0.5

Team Member Name: Isis Sanchez		**Initials: IS**
Date of Work Performed	**Description of Work Performed**	**Time Spent on Work Performed (HRS)**
3/19/2020	Created sample data for ASSETS, PARTS_PURCHASED, and ASSET_SUPPLIER tables	3
3/21/2020	Created JOIN queries	.5
3/22/2020	Made final revisions to schema and proofread	.5
3/22/2020	Proofread M3 before submitting	.3
4/1/2020	Conducted basic test queries	2
4/2/2020	Edited/Continued queries	0.5
4/3/2020	Edited Query 5	0.1
4/5/2020	Edited Query 3/Formatted M4	0.25
4/5/2020	Proofread M4	0.30
4/9/2020	Completed Visualization/ View	0.67
4/14/2020	Created Index (Not included in final document)	1
4/20/2020	Meeting with professor to discuss procedure	0.4
4/21/2020	Completed Procedure	1.25
4/26/2020	Proofread M5	0.25
4/29/2020	Completed client introduction/situation	1
4/30/2020	Complete slides and recordings for milestone 6	2
5/1/2020	Rerecorded some slides and cut down time for milestone 6	0.5
5/2/2020	Began writing executive summary/formatting milestone 7	0.5
5/3/2020	Finished formatting and proofreading m7	1
	Total Hours for Isis:	**31.77**

Nayan Tez Vankayalapati		**Initials: VNT**
Date of Work Performed	**Description of Work Performed**	**Time Spent on Work Performed (HRS)**
1/26/20	Created Intro paragraph for M1	0.5
1/27/20	Updated resume	0.5
1/27/20	Team meeting 1	1.25
1/30/20	Created team Resource histogram	1
2/10/20	Created and edited ERD attributes on gliffy	2
2/10/20	Made excel file for Data dictionary	0.5
2/10/20	Team meeting 2	1
2/14/20	Data dictionary creation	2
2/16/20	Proofread M2 before submission	0.25
3/5/20	Team meeting 3	1.5
3/20/20	Few failed queries	2
3/22/20	Proofread M3	0.5
3/25/20	2 basic queries	3
3/26/20	3 nested queries	2
3/28/20	Basic queries 6&7	2
4/4/20	Revised basic and nested Queries	2
4/4/20	Formated queries for submission	1.5
4/4/20	Proofread M4	0.5
4/20/20	Created view and tableau Visualization	1
4/23/20	Proofread M5	0.5
4/28/20	Made slides and voice threads for M6	1.5
5/1/20	Added screenshots for M7	0.5
5/3/20	Proofread M7	0.5
	Total hours for Nayan	**28.5**


Team 5 Resource Histogram
Number of Hours
45
40
35
30
25
20
15
10
5
0
Nayan
Mat
Isis
Varun
Janae
Team Members
Final Report
VoiceThread/Presentation
Meetings with Professor
Views/Visuals/Dashboard
Indexes
Triggers/Procedures
Transactions
Basic/Advanced Query Tests

APPENDIX C:

Meeting Agendas

Date & Time: 1/27/2020 1:30PM

Team Meeting for: (Atlas Co., Team 5)

Prepared by: VS

Client's Name: TBA for M1

Team members in attendance:

1. **(Chair) - Varun Shourie**
2. **Nayan Tez**
3. **Janae Lewis**
4. **Mat Brewer**
5. **Isis Sanchez**

Meeting Objective: Get the project off to an effective start by building a rapport with team members, reviewing team objectives/philosophies, and relegating all necessary tasks.

Agenda:

- Review of the necessary specifications for Milestone 1.
- Do icebreakers and introductions with all group members.
- Complete and finalize project-related documents (i.e. project charter/code of ethics, member introductions and team profile).
- Discussion of high level, abstract team member roles based on conation types.
- Develop a list of necessary tasks to wrap up Milestone 1 for all members.

Action Item	Assigned To	Due Date
Post headshots to introductory paragraphs.	All members	1/28/2020
Finalize team logo and slogan.	Varun	1/29/2020
Prepare Excel histogram resources.	Nayan	1/31/2020
Journal entries showing time worked on and tasks worked on.	All members	1/31/2020
Proofread the entire M1 document.	All members	2/1/2020

Time meeting ended: 1/27/2020 2:50PM

Date and time of next meeting: 2/3/2020 1:30PM

Initials: VS

Milestone 2 Agenda

Date & Time: 2/10/2020 1:30PM

Team Meeting for: (Atlas Co., Team 5)

Prepared by: MB

Client's Name: Franklin's Management Services Inc.

Team members in attendance:

1. **(Chair) - Mat Brewer**
2. **Nayan Tez**
3. **Janae Lewis**
4. **Varun Shourie**
5. **Isis Sanchez**

Meeting Objective: Touch base on milestone 2 objectives and walk through the business rules together as a group. Discuss what needs to be done and when tasks should be completed by.

Agenda:

- Review of the necessary specifications for Milestone 2.
- Work on the EERD together during the meeting.
- Assign roles for Milestone 2

Action Item	Assigned To	Due Date
Meet Professor Huang in office hours and get clarifications on assignment	Varun Shourie, Nayan Tez, Isis Sanchez	2/11/20
Start creating all entities/relationships on EERD	All Members	2/10/20
Finalize all entities, relationships, attributes on EERD	Mat Brewer, Isis Sanchez, Varun Shourie	2/13/20
Define all attributes as per requirements on the Data Dictionary	Nayan Tez, Janae Lewis	2/13/20
Journal Entries for Milestone 2	All Members	2/15/20
Proofread Milestone 2	All Members	2/15/20

Time meeting ended: 2/10/2020 2:30 PM

Date and time of next meeting: 2/17/2020 1:30PM

Initials: MB

Date & Time: 3/5, 11:00 AM

Team Meeting for: Team 5, Atlas Co.

Client's Name: Franklin's Management Services

Team members in attendance:

1. Varun Shourie **2. Isis Sanchez**

3. Nayan Tez **4. Mat Brewer**

5. Janae Lewis

Meeting Objective: Discuss and plan metadata specifications for all project database tables.

Agenda:

- Assign action items for creating the schema and drafting tables and the metadata code
- Revise milestone 2

Action Item	Assigned To	Due Date
Revise milestone 2- revise multivalued attributes on ERD	Varun Shourie, Isis Sanchez	3/5/20
Create schema	Janae Lewis, Varun Shourie	3/14/20
Identify order of table entry	Varun Shourie	3/16/20
Draft metadata code	Mat Brewer, Janae Lewis, Varun Shourie	3/20/20
Insert sample data with basic queries	Mat Brewer, Isis Sanchez, Nayan Tez	3/21/20
Complete journal entries	Mat Brewer, Isis Sanchez, Nayan Tez, Janae Lewis, Varun Shourie	3/21/20
Debug queries	Janae Lewis, Varun Shourie	3/22/20

Time meeting ended: 3/5, 11:45 AM

Date and time of next meeting: 3/16, 1:30 PM

Milestone 4 Agenda

Date & Time: 3/30/2020 1:30PM

Team Meeting for: (Atlas Co., Team 5)

Prepared by: IS

Client's Name: Franklin's Management Services Inc.

Team members in attendance:

1. **(Chair) - Isis Sanchez**
2. **Mat Brewer**
3. **Janae Lewis**
4. **Varun Shourie**

Meeting Objective: Touch base on milestone 4 objectives and walk through the business rules together as a group. Discuss what needs to be done and when tasks should be completed by.

Agenda:

- Review of the necessary specifications for Milestone 4
- Assign roles for Milestone 4

Action Item	Assigned To	Due Date
Update EERD / identify changes made	Varun Shourie, Mat Brewer, Isis Sanchez	4/1/20
Provide screenshots/create INSERT and NESTED statements, basic tests: 1, 2, 3, 6, 7 Advanced tests: 4-6	Nayan Tez	4/1/20
Provide screenshots/ create advanced tests: 1-3, 7-9	Janae Lewis, Varun Shourie, Mat Brewer, Isis Sanchez	4/3/20
Provide screenshots/ Basic tests: 4, 5, 8-10	Varun Shourie, Mat Brewer, Isis Sanchez	4/3/20
Journal Entries for Milestone 4	Nayan Tez, Janae Lewis, Varun Shourie, Mat Brewer, Isis Sanchez	4/4/20
Proofread Milestone 4	Nayan Tez, Janae Lewis, Varun Shourie, Mat Brewer, Isis Sanchez	4/4/20

Time meeting ended: 3/30/2020 2:36 PM
Date and time of next meeting: 4/6/2020 1:30PM

Initials: IS

Milestone 5 Agenda

Date & Time: 4/13/2020 1:30PM

Team Meeting for: (Atlas Co., Team 5)

Prepared by: IS

Client's Name: Franklin's Management Services Inc.

Team members in attendance:

1. **(Chair) - Isis Sanchez**
2. **Mat Brewer**
3. **Janae Lewis**
4. **Varun Shourie**

Meeting Objective: Touch base on milestone 5 objectives and walk through the business rules together as a group. Discuss what needs to be done and when tasks should be completed by.

Agenda:

- Review of the necessary specifications for Milestone 5
- Assign roles for Milestone 5

Action Item	Assigned To	Due Date
2 Transactions - Provide code and screenshots	Varun Shourie, Mat Brewer	4/22/20
Trigger and procedure	Janae Lewis, Isis Sanchez	4/22/20
Indexes	Nayan Tez, Janae Lewis, Varun Shourie, Mat Brewer, Isis Sanchez	4/22/20
Analytics Dashboard Overview	Nayan Tez, Janae Lewis, Varun Shourie, Mat Brewer, Isis Sanchez	4/22/20
Proofread Milestone 4	Nayan Tez, Janae Lewis, Varun Shourie, Mat Brewer, Isis Sanchez	4/26/20

Time meeting ended: 4/13/2020 2:22 PM
Date and time of next meeting: 4/27/2020 1:30PM

Initials: IS

Date & Time: 4/21/2020 12:00pm

Team Meeting for: Team 5

Client's Name: Franklin's Management Services

Team members in attendance:

1. Isis Sanchez
2. Varun Shourie
3. Mat Brewer
4. Nayan Tez

Team members absent:

1. Janae Lewis

Meeting Objective: Discuss and plan the specifications of the Milestone 6 presentation.

Agenda:

- Address all feedback from previous milestones
- Decide what important points your team will show in the presentation
- Create a draft of the presentation – which topic will be addressed on each slide
- Fix all code from previous milestones
- Go over M2 specs for client; and assign coding of the M2 specs.

Action Item	Assigned To	Due Date
Fix code for previous milestone	Varun Shourie, Isis Sanchez	4/26/20
Slides - Intro Team Member Slide(s)	Varun Shourie, Isis Sanchez, Mat Brewer, Nayan Tez, Janae Lewis	4/27/20
Slides 6, 7, & 19	Isis Sanchez (Tentative)	4/29/20
Slides 8, 9 & 10	Nayan Tez (Tentative)	4/29/20
Slides 11, 12, & 13	Varun Shourie (Tentative)	4/29/20
Slides 14, 15, & 16	Mat Brewer (Tentative)	4/29/20
Slides - 17 & 18	Janae Lewis (Tentative)	4/29/20
Finish coding M2 specifications.	Varun Shourie, Isis Sanchez, Mat Brewer, Nayan Tez, Janae Lewis	4/27/20
Compile a comprehensive .txt file	Mat Brewer	5/1/20

Time meeting ended: 1:15pm

Date and time of next meeting: 4/27/20 @ 11:00am

Milestone 7 Agenda

Date & Time: 4/27/2020 2:00PM

Team Meeting for: (Atlas Co., Team 5)

Prepared by: IS

Client's Name: Franklin's Management Services Inc.

Team members in attendance:

1. **(Chair) - Isis Sanchez**
2. **Mat Brewer**
3. **Janae Lewis**
4. **Varun Shourie**

Meeting Objective: Touch base on milestone 7 objectives and walk through the business rules together as a group. Discuss what needs to be done and when tasks should be completed by.

Agenda:

- Review of the necessary specifications for Milestone 7
- Assign roles for Milestone 7

Action Item	Assigned To	Due Date
Client Queries	Varun Shourie, Mat Brewer	4/28/20
Update data dictionary/EERD	Isis Sanchez	4/28/20
Executive Summary	Isis Sanchez, Varun Shourie	5/2/20
Client Introduction/Current Situation	Janae Lewis, Isis Sanchez	4/27/20
Screenshots (Queries, Tables, Visualizations)	Nayan Tez, Janae Lewis, Varun Shourie, Mat Brewer, Isis Sanchez	5/3/20

Time meeting ended: 4/27/2020 PM

Date and time of next meeting: None

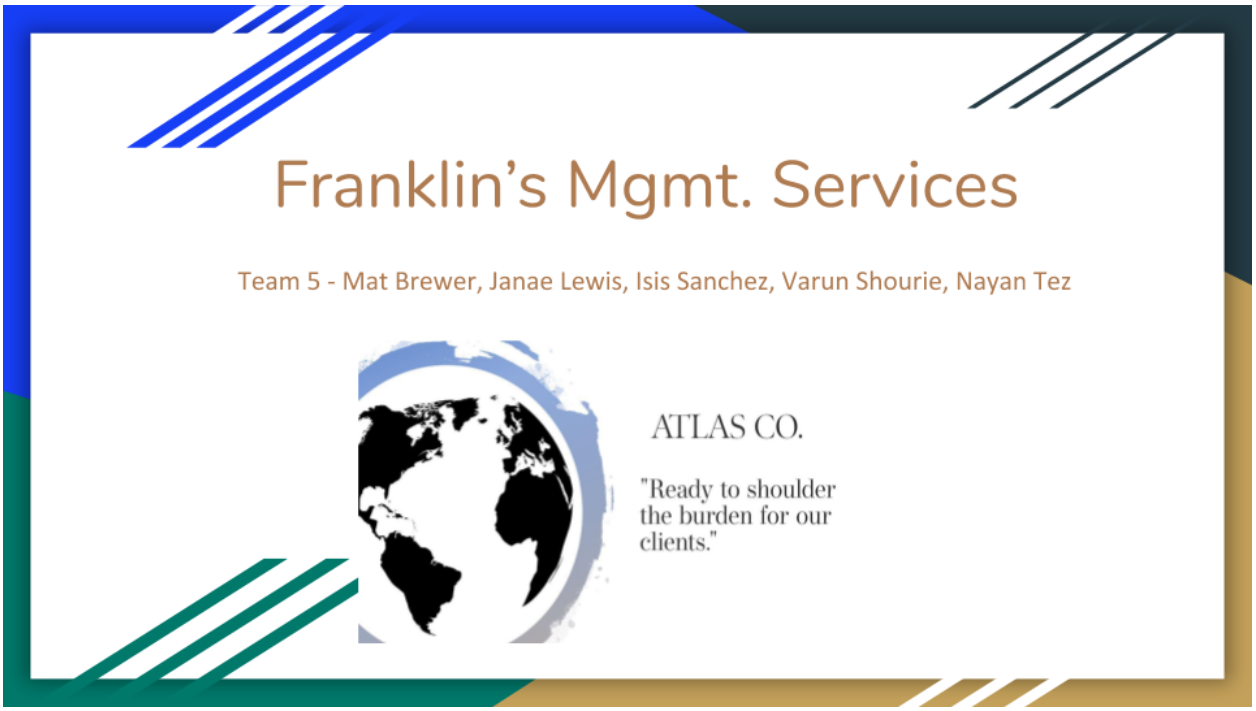
Initials: IS

APPENDIX D:

Presentation Screenshots

APPENDIX D:

Screenshots of your presentation (2 slides per page maximum). The client will look at these if unable to run/view your presentation.



Presentation Outline

1. Team Member Intros
2. Franklin's Management Services
3. Data Tables
4. Initial ERD
5. Revise ERD
6. Relational Model
7. Data Dictionary
8. Group By Query
9. Views
10. Trigger
11. Procedure
12. Transaction
13. Dashboard
14. Challenges
15. Lessons Learned

Team Member Intros - Mat Brewer

- CIS Major
- Conation Type: Quick Start
- Hobbies and Interests: hanging out with my friends, video games, weight lifting, and studying Japanese.
- Possible Career Goal: software developer or data scientist.

Team Member Intros - Janae Lewis

- CIS and Chinese double major
- Conation Type: Fact Finder
- Aspiring software developer with interests in front-end development and UI/UX design
- Enjoys traveling, reading, hiking, watching films, and
  Spending time with her friends and family

Team Member Intros - Varun Shourie

- Junior studying CIS
- Conation Type: Implementor/Facilitator
- Career goal: backend/database developer
- Spare time professional hobbies: Java/C# programming
- Spare time personal hobbies: hiking, running, video gaming

Team Member Intros - Nayan Tez

- Senior year majoring in Supply Chain Management and BDA.
- Conation Type: Fact Finder
- Aspire to work in the supply chain industry for automobile aftermarket.
- Hobbies include EFI tuning and playing Squash.

Team Member Intros - Isis Sanchez

- **Major(s):** Human Resources and Business Data Analytics
- **Conation Type**: Follow Through
- **Career Aspirations:** Human Resources Director or Compensation/Benefits Manager
- **Hobbies/Interests:** Video games, cats, coffee, and interior design
- **Fun Fact:** Spanish is my first language, relearning German

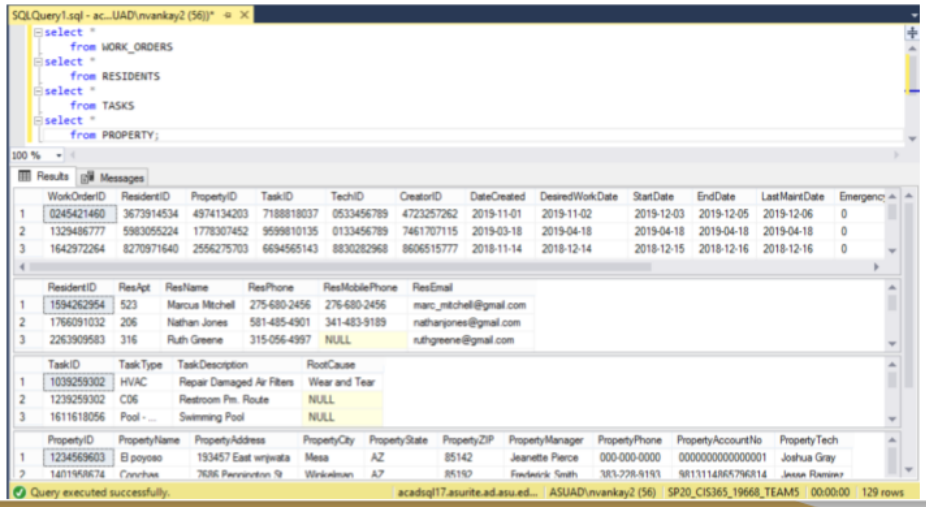
Franklin's Management Services

- **Medium sized company founded by Brick Franklin in 2002**
- **Located in Bedford Ohio, the company provides preventative maintenance services**
- **Keeping data current in Excel, which is affecting performance and payroll**
- **Require a database with the capacity to handle resident and general property inquiries**

Data Tables

- Based on our clients data needs, we planned on having 14 data tables for our database.
- We started populating the tables with about 30 tuples of sample data each.
- The work_orders table is in the centre of our design and is related to multiple tables in the database.
- The table has 5 foreign keys in it and when populating it with data, we had to make sure it matches with the data from the other tables

Data Tables

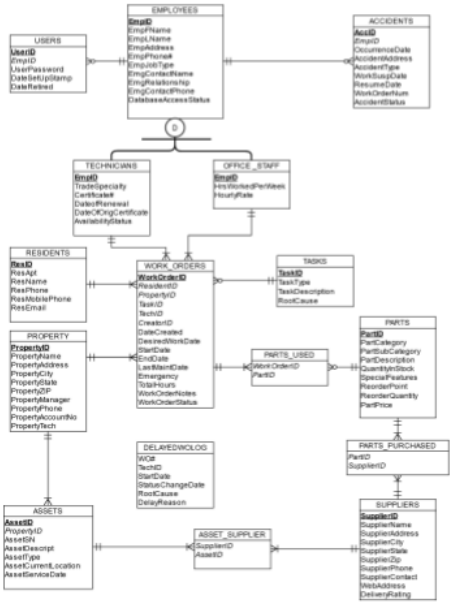


Initial ERD

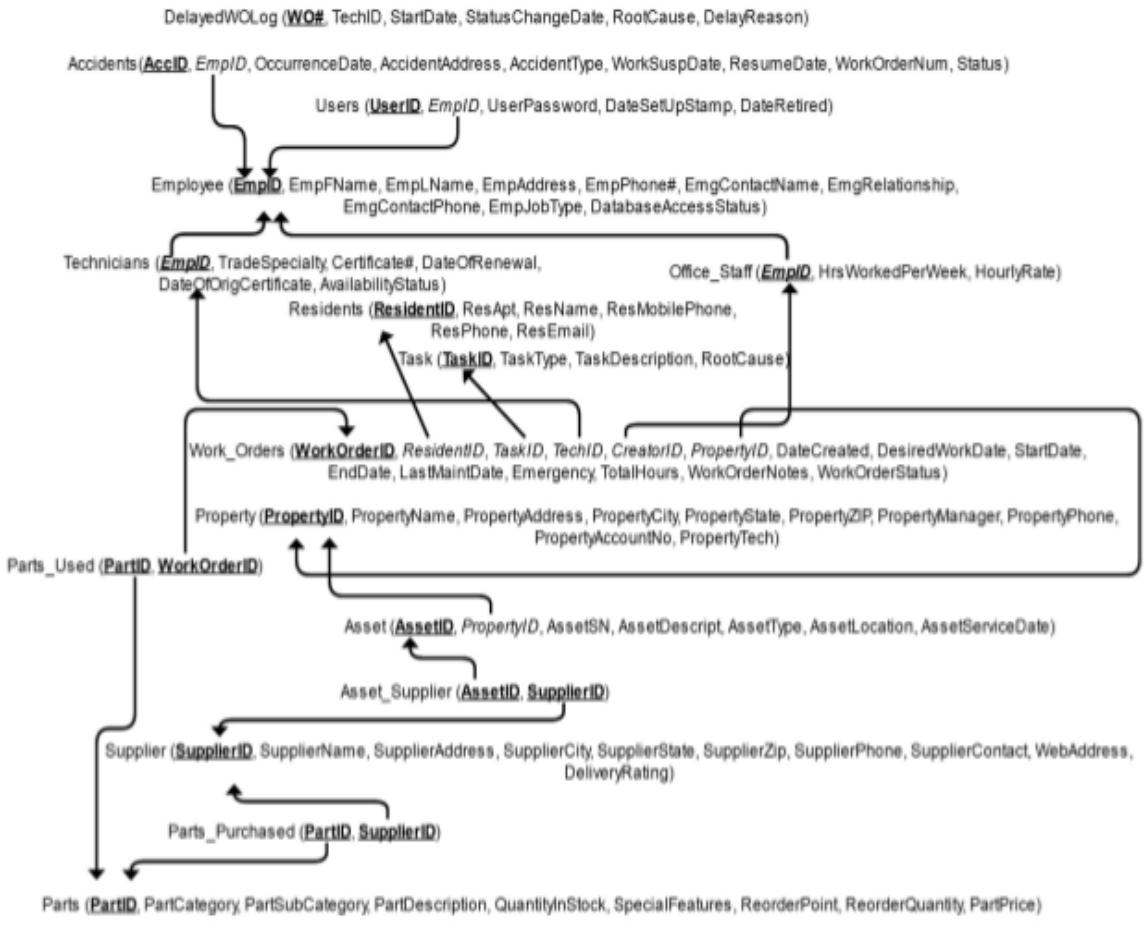
- After analysing the multiple aspects and data needs of our client's business, we made this initial ERD using Gliffy.

Revised ERD

- We then updated our ERD which had 14 tables initially by adding a parts_used table and one for delayed work orders.



Relational Model



Data Dictionary

Data Dictionary for Franklin's Management Services

Entity	Attribute	Definition
EMPLOYEES	EmpID	10-digit number uniquely identifying the employee, assigned by the company
EMPLOYEES	EmpFName	Employee's first name, character data, max-length 30 characters.
EMPLOYEES	EmpLName	Employee's last name, character data, max-length 30 characters.
EMPLOYEES	EmpAddress	Employee's home address, character data, max-length 100 characters. Listed as [(number) street name, city, state, ZIP code].
EMPLOYEES	EmpPhone#	Employee's listed phone number, character data, max length 12 characters, formatted as XXX-XXX-XXXX.
EMPLOYEES	EmpJobType	Distinguishes between office staff and technicians, character data, max length one character (e.g. "T" or "O")
EMPLOYEES	EmgContactName	Employee emergency contact's first and last name, character data, max-length 60 characters.
EMPLOYEES	EmgRelationship	Emergency contact's relationship to the employee, character data, max-length 30 characters.
EMPLOYEES	EmgContactPhone	Employee emergency contact's home/cell/office phone number, character data, max length 12 characters, formatted as XXX-XXX-XXXX.
EMPLOYEES	DatabaseAccessStatus	Defines an employee's access to the database, max length 8 characters. Values possible: "Active", "Inactive", or "N/A".
TECHNICIANS	EmpID	10-digit number uniquely identifying the employee, assigned by the company.
TECHNICIANS	TradeSpecialty	Technician's specific vocation, character data, max length 30 characters.
TECHNICIANS	Certificate#	ID of the technician's trade certificate, numeric data, max-length 20 digits.
TECHNICIANS	DateOfRenewal	Date of technician's trade certificate renewal, character data, max-length of 10 characters, formatted as yyyy-mm-dd.
TECHNICIANS	DateOfOrigCertificate	The date when the technician received their first certificate in their trade, used to derive work experience in years, character data, formatted as yyyy-mm-dd.
TECHNICIANS	AvailabilityStatus	Defines technician availability to be assigned to a work order, max 11 characters. Values possible: "Available" or "Unavailable".
OFFICE_STAFF	EmpID	10-digit number uniquely identifying the employee, assigned by the company.
OFFICE_STAFF	HrsWorkedPerWeek	Hours worked per week for office staff, numeric data, formatted as 0.00 with max-length of 4 digits & 2 decimal places.
OFFICE_STAFF	HourlyRate	Hourly pay rate for office staff, numeric data, formatted 0.00 with max-length of 5 digits & 2 decimal places.
ACCIDENTS	AccID	10-digit number uniquely identifying on-work employee accidents.
ACCIDENTS	EmpID	Employee's unique 10-digit number assigned by the company, not related to social security number or other personally identifiable information.
ACCIDENTS	OccurrenceDate	Date of the incident occurrence, character data, max length of 10 characters, formatted as yyyy-mm-dd.
ACCIDENTS	AccidentAddress	Location at which the incident occurred, character data, consists of a maximum of 100 characters. Also consists of [(number) street name, city, state, ZIP code].
ACCIDENTS	AccidentType	Type of accident the employee experienced on duty, character data, max-length of 30 characters.
ACCIDENTS	WorkSuspDate	Date in which the employee is suspended from work, character data, max length of 10 characters, formatted as yyyy-mm-dd.
ACCIDENTS	ResumeDate	Date in which the employee resumes work, character data, max length of 10 characters, formatted as yyyy-mm-dd.
ACCIDENTS	WorkOrderNum	10-digit number uniquely identifying the work order that the employee was working on, assigned randomly by the company.
ACCIDENTS	AccidentStatus	Defines the extent and gravity of the injury, character data, max 9 characters. Only three values possible: "emergency", "major", or "minor."
WORK_ORDERS	WorkOrderID	10-digit number uniquely identifying all work orders, assigned randomly by company.
WORK_ORDERS	ResidentID	10-digit number uniquely identifying each resident, assigned randomly by company.
WORK_ORDERS	PropertyID	10-digit number uniquely identifying each piece of property owned by the company.
WORK_ORDERS	TaskID	10-digit number uniquely identifying the tasks necessary to be completed for the company, assigned by the company.
WORK_ORDERS	TechID	10-digit number uniquely identifying the technician associated with the work order, assigned by the company. Same as EmpID in Technicians entity.
WORK_ORDERS	CreatorID	10-digit number uniquely identifying the employee which created the work order, assigned by the company. Same as EmpID in Office_Staff entity.
WORK_ORDERS	DateCreated	The date the work order was initially created by office staff, max length of 10 characters, formatted as yyyy-mm-dd.
WORK_ORDERS	DesiredWorkDate	Desired work date for the work order, character data, max length of 10 characters, formatted as yyyy-mm-dd.
WORK_ORDERS	StartDate	Date on which work starts in the work order, character data, max length of 10 characters, formatted as yyyy-mm-dd.
WORK_ORDERS	EndDate	End date of the work order, character data, max length of 10 characters, formatted as yyyy-mm-dd.
WORK_ORDERS	LastMaintDate	The date of the last maintenance activity for the work order, character data, max length of 10 characters, formatted as yyyy-mm-dd.
WORK_ORDERS	Emergency	Whether or not the work order is an emergency, numeric data, max length of 1 digit ("1" for yes,"0" for no, "-1" for yes due to misuse/neglect).
WORK_ORDERS	TotalHours	Total hours worked for the work order, numeric data, formatted 0.00 with max length of 5 digits and 2 decimal places.
WORK_ORDERS	WorkOrderNotes	Supplemental notes about the work order, character data, max length of 1000 characters.

Data Dictionary - Continued

WORK_ORDERS	WorkOrderStatus	Status of the work order, character data, max length of 9 characters, can hold 4 values: "assigned", "open", "delayed", or "completed".
RESIDENTS	ResID	10-digit number which uniquely identifies each resident in company properties, assigned by the company.
RESIDENTS	ResApt	Resident's apartment number, numeric data, max-length 3 digits (e.g. 101, 102, 203, etc.)
RESIDENTS	ResName	First and last name of the resident, character data, max-length 60 characters.
RESIDENTS	ResPhone	Resident's home phone number, character data, max-length 12 characters, formatted as XXX-XXX-XXXX.
RESIDENTS	ResMobilePhone	Resident's cell phone number, character data, max-length 12 characters, formatted as XXX-XXX-XXXX.
RESIDENTS	ResEmail	Resident's email to be used for electronic correspondence, character data, max-length 60 characters.
TASKS	TaskID	10-digit number uniquely identifying the tasks necessary to be completed for the company, assigned by the company.
TASKS	TaskType	The kind of task performed, character data, max-length 30 characters.
TASKS	TaskDescription	A basic explanation of the task, character data, max-length 100 characters.
TASKS	RootCause	An event that caused the need for a task to be completed, character data, max length of 1000 characters
PARTS	PartID	10-digit number uniquely identifying the part involved in maintenance, assigned by the company.
PARTS	PartCategory	The classification which best describes a part, character data, max-length 30 characters.
PARTS	PartSubCategory	A more detailed classification which a part belongs to, character data, max-length 30 characters.
PARTS	PartDescription	In depth explanation of a part's characteristics (such as brand and composition), character data, max-length 100 characters.
PARTS	QuantityInStock	Number of parts that are currently in stock, numeric data, max-length 3 digits.

PARTS	SpecialFeatures	Special characteristics of a part, character data, max length 100 characters.
PARTS	ReorderPoint	Number that identifies at what quantity of inventory a part should be re-ordered, numeric data, max-length 2 digits.
PARTS	ReorderQuantity	The amount of parts to reorder once the stock reaches the reorder point, numeric data, max-length 3 digits.
PARTS	PartPrice	Listed price of the part, numeric data, max-length 5-digits, formatted with 2 decimals (e.g. XX.XX or XXX.XX).
PARTS_USED	WorkOrderID	10-digit number uniquely identifying all work orders, assigned randomly by company.
PARTS_USED	PartID	10-digit number uniquely identifying the part involved in maintenance, assigned by the company.
PROPERTY	PropertyID	10-digit number uniquely identifying a property managed by the company, assigned by the company.
PROPERTY	PropertyName	Name of the property managed by the company, character data, max length 30 characters.
PROPERTY	PropertyAddress	Address of the property managed by the company, character data, max length 50 characters.
PROPERTY	PropertyCity	City where the property is located, character data, max length of 30 characters.
PROPERTY	PropertyState	State where the property is located, character data, max length 2 characters.
PROPERTY	PropertyZIP	ZIP code of the property address, character data, XXXXX or XXXXX-XXXX, max-length 10 characters.
PROPERTY	PropertyManager	Name of the person who manages the property, character data, max length 30 characters.
PROPERTY	PropertyPhone	Phone number attached to the property, character data, max length 12 characters, XXX-XXX-XXXX.
PROPERTY	PropertyAccountNo	Account number of the property holder, character data, max-length 16 characters.
PROPERTY	PropertyTech	Technician of the property to which he or she is assigned, character data, max-length 30 characters.
USERS	UserID	A character ID which uniquely identifies each user of the database, assigned by the company, max-length of 20 characters

Data Dictionary - Continued

USERS	EmpID	10-digit number uniquely identifying the employee, assigned by the company.
USERS	UserPassword	Password to the user's account in the database, character data, max-length 30 characters.
USERS	DateSetUpStamp	Date that the user was granted access into the database, character data, max-length 10 characters, formatted as yyyy-mm-dd
USERS	DateRetired	Date that the user's credentials were revoked, character data, max-length 10 characters, formatted as yyyy-mm-dd
PARTS_PURCHASED	PartID	10-digit number uniquely identifying the part involved in maintenance, assigned by the company.
PARTS_PURCHASED	SupplierID	10-digit number which uniquely identifies one of the company's suppliers, assigned by the company.
SUPPLIERS	SupplierID	10-digit number which uniquely identifies one of the company's suppliers, assigned by the company.
SUPPLIERS	SupplierName	Supplier's name, character data, max-length 30 characters.
SUPPLIERS	SupplierAddress	Supplier's address, character data, max-length 100 characters.
SUPPLIERS	SupplierCity	Supplier's city location, character data, max-length 30 characters.
SUPPLIERS	SupplierState	Supplier's state location, character data, max-length 2 characters.
SUPPLIERS	SupplierZIP	Supplier's ZIP code, character data, max-length 10 digits, XXXXX or XXXXX-XXXX.
SUPPLIERS	SupplierPhone	Supplier's phone number, character data, max-length 10 digits, (XXX)-XXX-XXXX.
SUPPLIERS	SupplierContact	Name of the supplier's contact person, character data, max-length 60 characters.
SUPPLIERS	WebAddress	Supplier's website address (aka URL), character data, max-length 200 characters.
SUPPLIERS	DeliveryRating	Supplier's quality of delivery, character data, max-length 30 characters. Can assume values such as "late, damaged, on-time, defaulted, etc."
ASSETS	AssetID	10-digit number which uniquely identifies one of the assets under the company's control, assigned by the company.

ASSETS	PropertyID	10-digit number uniquely identifying a property managed by the company, assigned by the company.
ASSETS	AssetSN	Serial number of the asset under company control, assigned by the manufacturer of the asset, maximum length of 20 characters.
ASSETS	AssetDescript	Detailed explanation of the asset, character data, max-length 100 characters.
ASSETS	AssetType	Type of the asset under company control, character data, max-length 60 characters.
ASSETS	AssetLocation	Current location of the asset, character data, max-length 60 characters. For example, this is usually a room or a specific building.
ASSETS	AssetServiceDate	Date of the maintenance service performed upon the asset, character data, max-length 10 characters, formatted as yyyy-mm-dd.
ASSET_SUPPLIER	SupplierID	10-digit number which uniquely identifies one of the company's suppliers, assigned by the company.
ASSET_SUPPLIER	AssetID	10-digit number which uniquely identifies one of the assets under the company's control, assigned by the company.
DelayedWOLog	WO#	10-digit number uniquely identifying all work orders, assigned randomly by company.
DelayedWOLog	StartDate	Date on which work starts in the work order, character data, max length of 10 characters, formatted as yyyy-mm-dd.
DelayedWOLog	TechID	10-digit number uniquely identifying the technician associated with the work order, assigned by the company. Same as EmpID in Technicians entity and TechID in Work_Orders entity.
DelayedWOLog	StatusChangeDate	Date on which the work order status was changed, character data, max-length 10 characters, formatted as yyyy-mm-dd.
DelayedWOLog	RootCause	An event that caused the need for a task to be completed, character data, max length of 1000 characters. Same as RootCause in Tasks entity.
DelayedWOLog	DelayReason	Supplemental notes relating to cause for delay, character data, max length of 1000 characters.

Group By Query

```
Milestone4.sql - ac...SUAD\vshourie (63))*
46  /* Advanced Query #1
47  List the number of incidences of each accident type with more than one accident */
48  SELECT AccidentType, COUNT(AccidentType) AS "Number of Accidents"
49      FROM ACCIDENTS
50          GROUP BY AccidentType
51              HAVING COUNT(AccidentType) > 1
52  ORDER BY COUNT(AccidentType) DESC;
```

Results | Messages

	AccidentType	Number of Accidents
1	Car crash	10
2	Fall from ladder	2
3	Fell down stairs	2

- Group by paves the way for visualizations!
- Quantitative figures!
- Aids in decision-making.

Views

```
Milestone4.sql - ac...UAD\vshourie (146))*   IC09.sql - acadsql1...UAD\vshourie (160))
29
30  /* Advanced Test #9 - Views (Taken from the "RESULTS" section of Milestone 2)
31  "Residents who have had maintenance work completed in the past 6 months." */
32  CREATE VIEW ResidentsRecentMaintenance AS
33      (SELECT R.ResidentID AS "Resident ID",
34              R.ResApt AS "Resident's Apartment #",
35              R.ResName AS "Resident's Name",
36              W.WorkOrderID AS "Work Order ID",
37              W.LastMaintDate AS "Date Last Maintenance Occurred"
38          FROM RESIDENTS R JOIN WORK_ORDERS W
39              ON R.ResidentID = W.ResidentID
40                  WHERE DATEDIFF(MONTH,W.LastMaintDate,GETDATE()) <= 6);
41  SELECT *
42      FROM ResidentsRecentMaintenance;
```

100 % | Results | Messages

	Resident ID	Resident's Apartment #	Resident's Name	Work Order ID	Date Last Maintenance Occurred
1	3673914534	410	Sophia Jackson	0245421460	2019-12-06
2	2263909583	316	Ruth Greene	1858869730	2020-08-22
3	9003459567	834	Zachary Marsh	2393881879	2020-04-07
4	5252053437	628	Brandon Pena	7174747431	2020-12-06
5	2640852291	766	Jennifer Smith	7491160348	2020-02-05
6	3498751707	286	Kristina Rangel	8893160795	2020-01-06

Query... | acadsql17.asurite.ad.asu.ed... | ASUAD\vshourie (146) | SP20_CIS365_19668_TEAM5 | 00:00:00 | 6 rows

- **Views update as time progresses, making it very powerful.**
- **As per the query results, this query has powerful business applications.**
- **Financial, customer service, marketing, etc.**

Trigger (WARNING: very complex).

/* 1. (10) TRIGGER: When a new work order is created (inserted), automatically assign the first available technician to do the job; assign a tech with the appropriate tech specialty (if the task is plumbing, don't assign an electrician).*/

```
ALTER TABLE TECHNICIANS                    -- Adds
    ADD AvailabilityStatus varchar(11); -- We as
UPDATE TECHNICIANS                              --
    SET AvailabilityStatus = 'Available'     --
        WHERE EmpID IN
            (SELECT TechID
                FROM WORK_ORDERS
                    WHERE EndDate IS NOT NULL);
```

	EmpID	TradeSpecialty	Certificate#	DateOfRenewal	DateOfOrigCertificate	AvailabilityStatus
1	0123456789	HVAC	77821236	2021-10-10	2019-10-10	Available
2	0133456789	Pools	75288777	2022-01-23	2000-05-15	Available
3	0233456789	Landscaping	97767240	2020-11-28	2008-11-28	Available
4	0333456789	Carpentry	38336214	2020-05-28	2015-05-28	Available
5	0433456789	Landscaping	49066751	2020-05-16	2006-05-16	Available

```
INSERT INTO TASKS
    VALUES('1039259302','HVAC','Repair Damaged Air Filters', 'Wear and Tear');
SELECT * FROM TASKS;
```

	TaskID	TaskType	TaskDescription	RootCause
1	1039259302	HVAC	Repair Damaged Air Filters	Wear and Tear

```
/* Creating the trigger */
CREATE TRIGGER TechniciansWOInsert
ON WORK_ORDERS
AFTER INSERT
AS
    UPDATE WORK_ORDERS
        SET TechID =
            (SELECT TOP 1 EmpID
                    FROM TECHNICIANS
                        WHERE TradeSpecialty IN
                            (SELECT TaskType
                                FROM TASKS
                                    WHERE TaskID IN
                                        (SELECT TaskID
                                            FROM inserted)
                                AND AvailabilityStatus = 'Available'))
            WHERE WorkOrderID IN
                (SELECT WorkOrderID
                    FROM inserted);
    UPDATE TECHNICIANS
        SET AvailabilityStatus = 'Unavailable'
            WHERE EmpID IN
                (SELECT TechID
                    FROM WORK_ORDERS
                        WHERE WorkOrderID IN
                            (SELECT WorkOrderID
                                FROM inserted));
```

Trigger cont'd.

/* 1. (10) TRIGGER: When a new work order is created (inserted), automatically assign the first available technician to do the job; assign a tech with the appropriate tech specialty (if the task is plumbing, don't assign an electrician).*/

BEFORE Test

```
SELECT *
    FROM WORK_ORDERS
        WHERE WorkOrderID = '8437549859';
SELECT * FROM TECHNICIANS;
```

AFTER Test

```
INSERT INTO WORK_ORDERS
    VALUES ('8437549859', '6393256990', '9534546401', '1039259302', NULL, '3663027200', '2020-04-28', '2020-04-28',
    '2020-05-05', NULL, NULL, '0', 23.38, 'Damaged Filter', 'Completed');
SELECT * FROM WORK_ORDERS
    WHERE WorkOrderID = '8437549859';
SELECT * FROM TECHNICIANS;
```

BEFORE RESULTS

	EmpID	TradeSpecialty	Certificate#	DateOfRenewal	DateOfOrigCertificate	AvailabilityStatus
1	0123456789	HVAC	77821236	2021-10-10	2019-10-10	Available

WorkOrderID	ResidentID	PropertyID	TaskID	TechID	CreatorID	DateCreated	DesiredWorkDate	StartDate	EndDate	LastMaintDate	Emergency	TotalHours	WorkOrderNotes

AFTER RESULTS

	EmpID	TradeSpecialty	Certificate#	DateOfRenewal	DateOfOrigCertificate	AvailabilityStatus
1	0123456789	HVAC	77821236	2021-10-10	2019-10-10	Unavailable

	WorkOrderID	ResidentID	PropertyID	TaskID	TechID	CreatorID	DateCreated	DesiredWorkDate	StartDate	EndDate	LastMaintDate	Emergency	TotalHours	Work
1	8437549859	6393256990	9534546401	1039259302	0123456789	3663027200	2020-04-28	2020-04-28	2020-05-05	NULL	NULL	0	23.38	Dam

Procedure

```
/* PROCEDURE: Creating procedure AddDelayedWO to update DelayedWOLog  */
CREATE PROCEDURE AddDelayedWO
        @WorkOrderID    char(10),
        @DelayReason    varchar(1000)
    AS
                INSERT INTO DelayedWOLog(WO#, StartDate, TechID, StatusChangeDate, RootCause, DelayReason)
                    VALUES (@WorkOrderID,
                                (SELECT StartDate
                                    FROM WORK_ORDERS
                                        WHERE WorkOrderID = @WorkOrderID
                                        AND (WorkOrderStatus = 'Delayed' OR WorkOrderStatus = 'delayed')),
                                (SELECT TechID
                                    FROM WORK_ORDERS
                                        WHERE WorkOrderID = @WorkOrderID
                                        AND (WorkOrderStatus = 'Delayed' OR WorkOrderStatus = 'delayed')),
                                GETDATE(),
                                (SELECT RootCause
                                    FROM TASKS
                                        WHERE TaskID IN
                                            (SELECT TaskID
                                                FROM WORK_ORDERS
                                                    WHERE WorkOrderID = @WorkOrderID
                                                    AND (WorkOrderStatus = 'Delayed' OR WorkOrderStatus = 'delayed'))),
                                @DelayReason);

EXEC AddDelayedWO 2393881879, 'Improper tools';
EXEC AddDelayedWO 9375544818, 'Missing part';

SELECT *
    FROM DelayedWOLog;
```

SQLQuery1.sql - a...UAD\mbrewer7 (59))*

```
/* PROCEDURE: Creating table DelayedWOLog to log when WorkOrder Status is changed to delayed  */
CREATE TABLE DelayedWOLog (
WO#                 char(10),
StartDate           date,
TechID              char(10),
StatusChangeDate    datetime        DEFAULT GETDATE(),
RootCause           varchar(1000),
DelayReason         varchar(1000));
```

100 %

Results | Messages

	WO#	StartDate	TechID	StatusChangeDate	RootCause	DelayReason
1	2393881879	2020-04-06	0433456789	2020-04-26 12:01:43.417	car crashed into it	Improper tools
2	9375544818	2002-07-27	2051366864	2020-04-26 12:01:46.580	NULL	Missing part

Transactions

```
BEGIN TRANSACTION;

Insert into SUPPLIERS values ('1234567890', 'Daniel Jones', '11198 East Camina Plata', 'Queen Creek', 'AZ', '85142', '480-924-7013', 'Rock Lee', 'danieljones@gmail.com', 'on-time');

Insert into PARTS values('1234567891', 'Pumps', '1" Diaphragm Pump', 'Stainless Steel w/ Santoprene Diaphragm', '5', NULL, '2', '2', 450.00);
Insert into PARTS values('1234567892', 'Pumps', '1.5" Diaphragm Pump', 'Stainless Steel w/ Santoprene Diaphragm', '5', NULL, '2', '2', 500.00);
Insert into PARTS values('1234567893', 'Pumps', '3" Diaphragm Pump', 'Stainless Steel w/ Santoprene Diaphragm', '5', NULL, '3', '3', 550.00);

Insert into PARTS_PURCHASED values('1234567891', '1234567890');
Insert into PARTS_PURCHASED values('1234567892', '1234567890');
Insert into PARTS_PURCHASED values('1234567893','1234567890');

--AFTER
select *
    from SUPPLIERS
        where SupplierID = '1234567890';
select *
    from PARTS
        where PartID = '1234567891'
        or PartID = '1234567892'
        or PartID = '1234567893';
select *
    from PARTS_PURCHASED
        where PartID = '1234567891'
        or PartID = '1234567892'
        or PartID = '1234567893';

ROLLBACK; --incase there is an error
COMMIT;
```

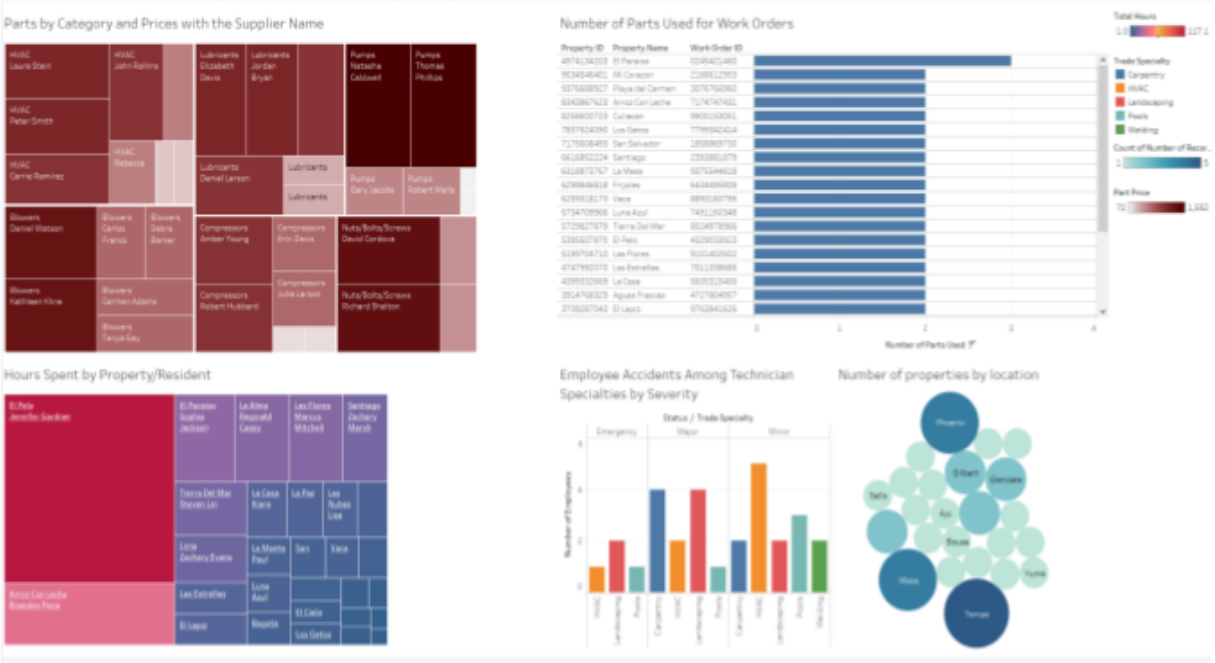
Results | Messages | Execution plan

	SupplierID	SupplierName	SupplierAddress	SupplierCity	SupplierState	SupplierZip	SupplierPhone	SupplierContact	WebAddress	DeliveryRating
1	1234567890	Daniel Jones	11198 East Camina Plata	Queen Creek	AZ	85142	480-924-7013	Rock Lee	danieljones@gmail.com	on-time

	PartID	PartCategory	PartSubCategory	PartDescription	QuantityInStock	SpecialFeatures	ReorderPoint	ReorderQuantity	PartPrice
1	1234567891	Pumps	1" Diaphragm Pump	Stainless Steel w/ Santoprene Diaphragm	5	NULL	2	2	450.00
2	1234567892	Pumps	1.5" Diaphragm Pump	Stainless Steel w/ Santoprene Diaphragm	5	NULL	2	2	500.00
3	1234567893	Pumps	3" Diaphragm Pump	Stainless Steel w/ Santoprene Diaphragm	5	NULL	3	3	550.00

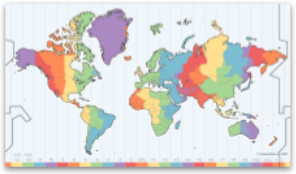
	PartID	SupplierID
1	1234567891	1234567890
2	1234567892	1234567890
3	1234567893	1234567890

Dashboard



Can show a myriad of stories with our database.

Challenges



- Cross-time zone collaboration - stick to a routine
- Automation of the database (triggers/procedures)
  - Treat it like an iterative process - tweak your design slightly.
- Challenge of vagueness which comes with the real-world
  - Treat your professor like a client!
- Losing a team member - everyone stepped up.

Lessons Learned

- Test your changes to the database on a separate table.
- Knowing how to debug your database is more important than being able to program your database!
- Do not take the easy way out! Client satisfaction and functionality is the ultimate goal of database design