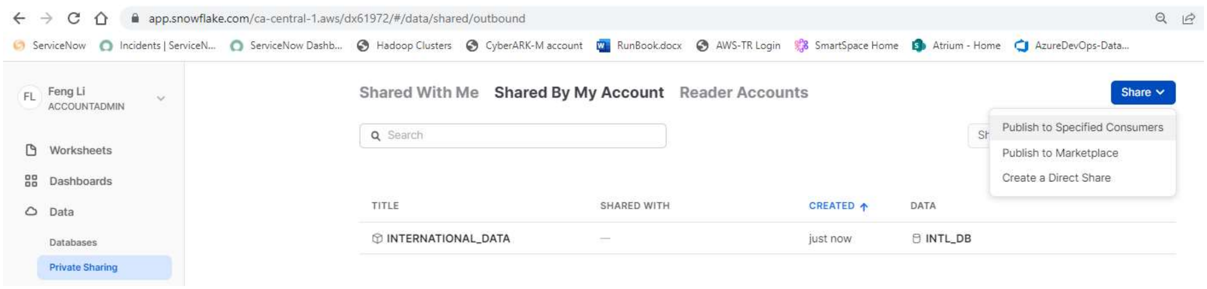
Badge 3

Thursday, December 15, 2022 11:26 PM

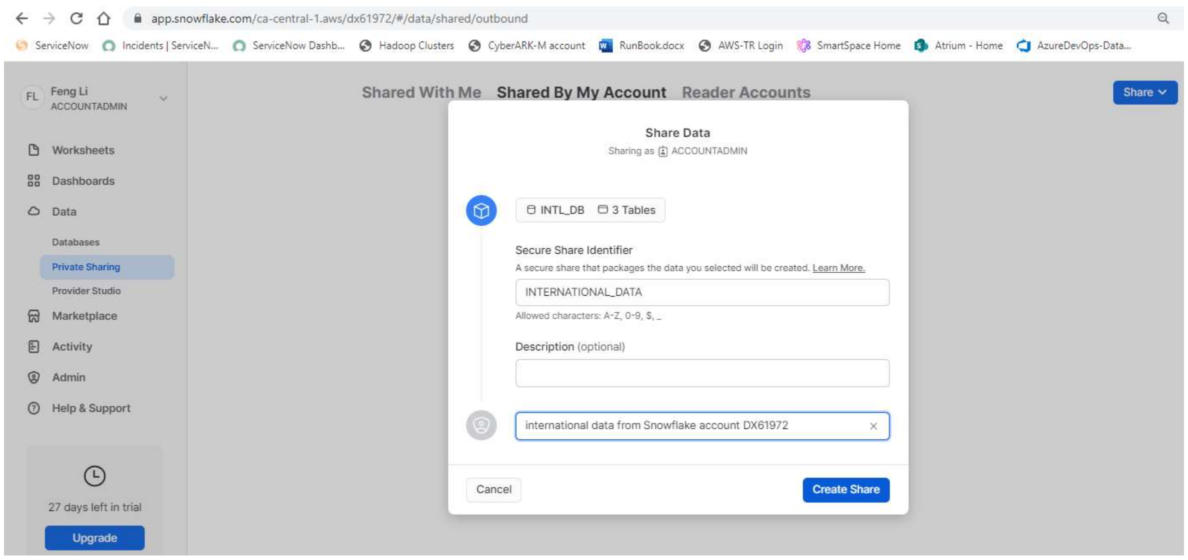
https://learn.snowflake.com/courses/course-v1:snowflake+ESS-SMEW+C/courseware/160ccfe9093e43b78c86233845deacdc/4a551c8828224d7bb158afe454ce4342/

- For advanced tutorials, go to quickstarts.snowflake.com
- For instructor-led training go to training.snowflake.com
- For documentation go to docs.snowflake.com
- For forum discussions and technical support go to community.snowflake.com

From <https://learn.snowflake.com/courses/course-v1:snowflake+ESS-SMEW+C/courseware/160ccfe9093e43b78c86233845deacdc/bce698eda67444798289e4be15a1c54a/?child=first>



TITLE	SHARED WITH	CREATED	DATA
INTERNATIONAL_DATA	—	just now	INTL_DB



Share Data

Sharing as ACCOUNTADMIN

INTL_DB 3 Tables

Secure Share Identifier

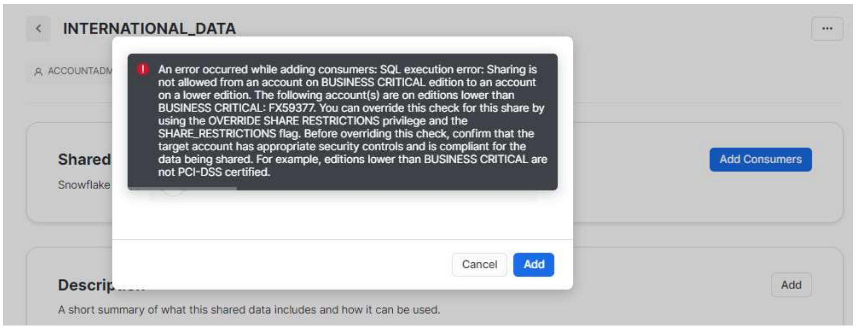
A secure share that packages the data you selected will be created. Learn More.

INTERNATIONAL_DATA

Allowed characters: A-Z, 0-9, $, _

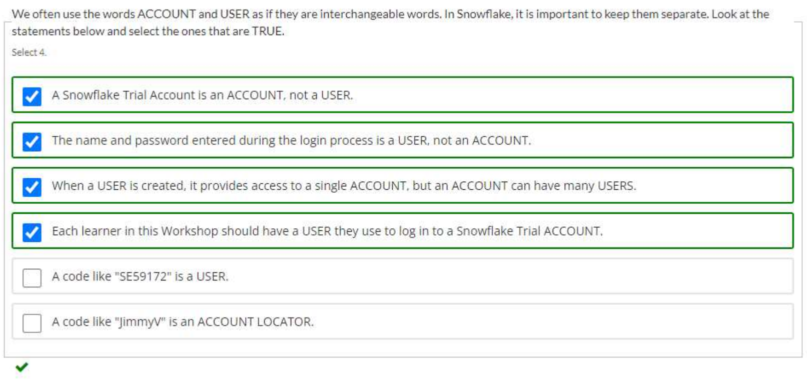
Description (optional)

international data from Snowflake account DX61972



An error occurred while adding consumers: SQL execution error: Sharing is not allowed from an account on BUSINESS CRITICAL edition to an account on a lower edition. The following account(s) are on editions lower than BUSINESS CRITICAL: FX59377. You can override this check for this share by using the OVERRIDE SHARE RESTRICTIONS privilege and the SHARE_RESTRICTIONS flag. Before overriding this check, confirm that the target account has appropriate security controls and is compliant for the data being shared. For example, editions lower than BUSINESS CRITICAL are not PCI-DSS certified.

A short summary of what this shared data includes and how it can be used.

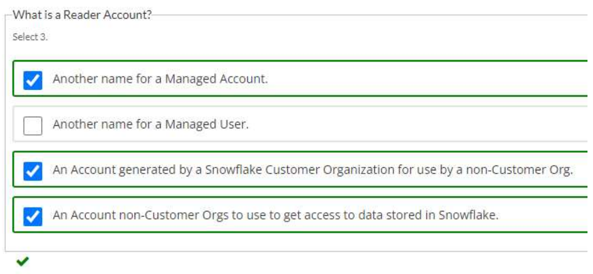


We often use the words ACCOUNT and USER as if they are interchangeable words. In Snowflake, it is important to keep them separate. Look at the statements below and select the ones that are TRUE.

Select 4.

- [x] A Snowflake Trial Account is an ACCOUNT, not a USER.
- [x] The name and password entered during the login process is a USER, not an ACCOUNT.
- [x] When a USER is created, it provides access to a single ACCOUNT, but an ACCOUNT can have many USERS.
- [x] Each learner in this Workshop should have a USER they use to log in to a Snowflake Trial ACCOUNT.
- [ ] A code like "SE59172" is a USER.
- [ ] A code like "JimmyV" is an ACCOUNT LOCATOR.

✓

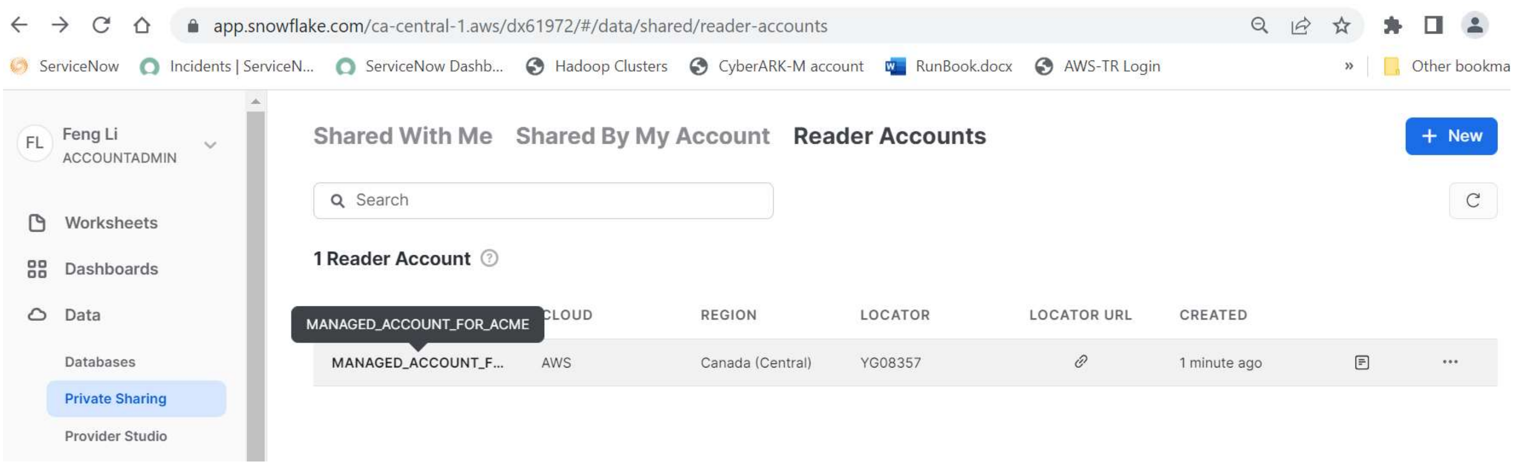


What is a Reader Account?

Select 3.

- [x] Another name for a Managed Account.
- [ ] Another name for a Managed User.
- [x] An Account generated by a Snowflake Customer Organization for use by a non-Customer Org.
- [x] An Account non-Customer Orgs to use to get access to data stored in Snowflake.

✓



app.snowflake.com/ca-central-1.aws/dx61972/#/data/shared/reader-accounts

Shared With Me | Shared By My Account | **Reader Accounts**

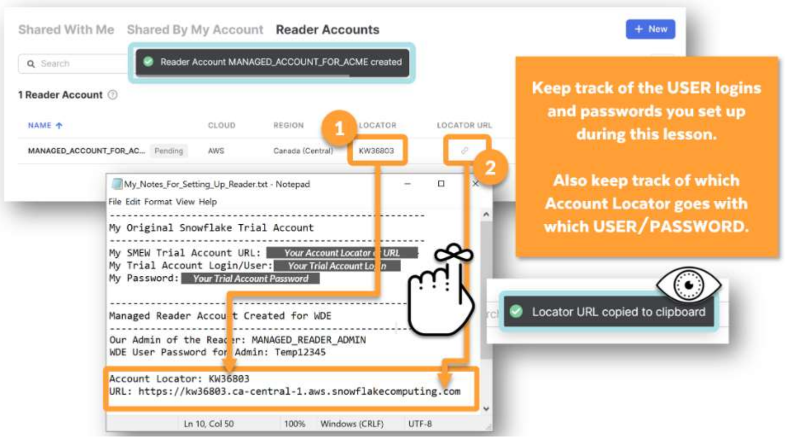
1 Reader Account

MANAGED_ACCOUNT_FOR_ACME

NAME	CLOUD	REGION	LOCATOR	LOCATOR URL	CREATED
MANAGED_ACCOUNT_F...	AWS	Canada (Central)	YG08357		1 minute ago

Account: managed_account_for_acme
User: managed_reader_admin
Pwd: Temp12345
Locator: YG08357
Reader Account url: https://yg08357.ca-central-1.aws.snowflakecomputing.com

My trial account: https://app.snowflake.com/ca-central-1.aws/dx61972/#/data/shared/reader-accounts



Reader Account MANAGED_ACCOUNT_FOR_ACME created

NAME	CLOUD	REGION	LOCATOR	LOCATOR URL
MANAGED_ACCOUNT_FOR_AC... Pending	AWS	Canada (Central)	KW36803	

Keep track of the USER logins and passwords you set up during this lesson.

Also keep track of which Account Locator goes with which USER/PASSWORD.

Locator URL copied to clipboard

My_Notes_For_Setting_Up_Reader.txt - Notepad

```
------------------------------------------------------
My Original Snowflake Trial Account
------------------------------------------------------
My SMEW Trial Account URL:  Your Account Locator or URL
My Trial Account Login/User: Your Trial Account Login
My Password: Your Trial Account Password

------------------------------------------------------
Managed Reader Account Created for WDE
------------------------------------------------------
Our Admin of the Reader: MANAGED_READER_ADMIN
WDE User Password for Admin: Temp12345

Account Locator: KW36803
URL: https://kw36803.ca-central-1.aws.snowflakecomputing.com
```

create database fengdb;
create schema fengdb.fengschema;
create or replace table fengtbl (

```
    my_dates varchar
);
insert into fengtbl (my_dates) values ('4/30/2022 11:13 pm');
--insert into fengtbl (my_dates) values (to_timestamp('4/30/2022 11:13 pm', 'MM/DD/YYYY hh12:mi ampm'));
select * from fengtbl;


create or replace table fengtbl2 (
    my_text timestamp
);
--insert into fengtbl2 (my_text) values('4/20/2022 11:13 pm');
select * from fengtbl2;

insert into fengtbl2 (my_text) select to_timestamp(my_dates, 'MM/DD/YYYY hh12:mi ampm') from fengtbl;
--insert into fengtbl2 select my_dates from fengtbl;
select * from fengtbl2;

desc table fengtbl;



--------------
execute immediate $$
declare
    x int default 0;
begin
    create or replace table xxx(i int);
    loop
        x := x + 1;
        execute immediate 'insert into xxx(i) values (' || x || ')';
        if (x >= 20) then break;
        end if;
    end loop;
    return x;
end;
$$;
select * from xxx;

--('2021-01','2021-02','2021-03')
execute immediate $$
declare
    x timestamp default '2021-01';
begin
    create or replace table yyy(i timestamp);
    loop
        x := x + 1;
        execute immediate 'insert into xxx(i) values (' || x || ')';
        if (x >= 20) then break;
        end if;
    end loop;
    return x;
end;
$$;
select * from yyy;


set mydate='2021-01';
select to_date($mydate);

set my_variable=10;

show variables;
set my_variable='example';

set (var1, var2, var3)=(10, 20, 30);

select $var1;

select [ 'Alberta', 'manitoba' ] as province;



create or replace table some_date (id integer, mydate date);
insert into some_date (id, mydate) values
  (1, '2021-01-01'),
  (2, '2021-02-01'),
  (3, '2021-03-01');
select * from some_date;

create or replace table some_date2 (id integer, newdate date);


execute immediate $$
```

```
declare
  x int default 0;
begin
  loop
    x := x + 1;
    if (x > 3) then break;
    end if;
    execute immediate 'insert into some_date2 select id, mydate from some_date where id = ' || x;
  end loop;
end;
$$
;
select * from some_date2;

--------------- above works -----------------
-- execute immediate 'insert into some_data2 select id, name into :id, :name from some_data where id = :x';
-- execute immediate 'insert into some_data2 select id, name from some_data where id = :x';
--    execute immediate 'insert into zzz(i) values (' || x || ')';


execute immediate $$
declare
  x int default 0;
  mydate date;
begin
  loop
    x := x + 1;
    if (x > 3) then break;
    end if;
    execute immediate 'insert into some_date2 select id, mydate from some_date where id = ' || x;
  end loop;
end;
$$
;
select * from some_date2;

create or replace table some_date (id integer, mydate date);
insert into some_date (id, mydate) values
  (1, '2021-01-01'),
  (2, '2021-02-01'),
  (3, '2021-03-01');
select * from some_date;

create or replace table some_date2 (id integer, newdate date);


execute immediate $$
declare
  x int default 0;
  mydate date;
begin
  loop
    x := x + 1;
    if (x > 3) then break;
    end if;
    mydate = select mydate from some_date where id = :x;
    execute immediate 'insert into some_date2 select id, mydate from some_date where id = ' || x;
  end loop;
end;
$$
;
select * from some_date2;

------------------

create or replace table some_date (id integer, mydate date);
insert into some_date (id, mydate) values
  (1, '2021-01-01'),
  (2, '2021-02-01'),
  (3, '2021-03-01');
select * from some_date;

create or replace table data_tbl (price number(12, 2), mydate date);
insert into data_tbl (price, mydate) values
    (11.11, '2021-01-01'),
    (22.22, '2021-02-01'),
    (33.33, '2021-03-01'),
    (44.44, '2021-04-01'),
    (55.55, '2021-05-01'),
    (66.66, '2021-06-01');
select * from data_tbl;

create or replace table new_data_tbl (price number(12, 2), mydate date);
```

```
select * from new_data_tbl;
select mydate from some_date;
select id from some_date;

execute immediate $$
declare
  x int default 0;
  mydate date;
begin
  loop
    x := x + 1;
    if (x > 3) then break;
    end if;
    mydate = select mydate from some_date where id = :x;
    execute immediate 'insert into some_date2 select id, mydate from some_date where id = ' || x;
  end loop;
end;
$$;

create or replace procedure for_loop_date_over_cursor()
returns text
language sql
as
$$
declare
    total_date text;
    total_date1 text;
    mydates cursor for select id from some_date;
begin
  loop
    x := x + 1;
    if (x > 3) then break;
    end if;
    mydate = select mydate from some_date where id = :x;
    execute immediate 'insert into new_data_tbl select price, mydate from data_tbl where mydate = ' || mydate;
  end loop;
  return 'success';
end;
$$;
call for_loop_date_over_cursor();
select * from new_data_tbl;
        execute immediate 'insert into new_data_tbl select price, mydate from data_tbl where mydate = ' || target_date;
total_date := total_date || target_date;

--
create or replace table invoices (price number(12, 2));
insert into invoices (price) values
    (11.11),
    (22.22);
create or replace procedure for_loop_over_cursor()
returns float
language sql
as
$$
declare
    total_price float;
    c1 cursor for select price from invoices;
begin
    total_price := 0.0;
    open c1;
    for rec in c1 do
        total_price := total_price + rec.price;
    end for;
    close c1;
    return total_price;
end;
$$
;
call for_loop_over_cursor();


create or replace table some_date (id integer, mydate text);
insert into some_date (id, mydate) values
  (1, '2021-01-01'),
  (2, '2021-02-01'),
  (3, '2021-03-01');
select * from some_date;


execute immediate $$
declare
  target_date date;
  mydates cursor for select mydate from some_date;
```

```
begin
  open mydates;
  for myrow in mydates do
      target_date := myrow.mydate::date;
      insert into new_data_tbl select price, mydate from data_tbl where mydate = :target_date;
  end for;
  close mydates;
  return 'success';
end;
$$;

-- above works ---
      execute immediate 'insert into new_data_tbl select price, mydate from data_tbl where mydate = $target_date';

select to_date('2021-02-01');
select * from new_data_tbl;
truncate table new_data_tbl;


create or replace procedure inserting_data()
returns text
language sql
as
$$
declare
  mydates cursor for select mydate from some_date;
  target_date date;
begin
  open mydates;
  for myrow in mydates do
      target_date := myrow.mydate::date;
      insert into new_data_tbl select price, mydate from data_tbl where mydate = :target_date;
  end for;
  return 'success';
end;
$$
;
call inserting_data();
select * from new_data_tbl;

select * from data_tbl where mydate in (select mydate from some_date);

----------------------------------------------
create or replace table student (name text, mark int);
insert into student values
    ('Tom', 80),
    ('John', 70);
create or replace table tmp (name text);

create or replace procedure example_if(flag integer, mark integer)
returns varchar
language sql
as
$$
begin
    if (flag = 1) then
        insert into tmp select name from student where mark >= :mark;
        return 'Success: flag is 1';
    elseif (flag = 0) then
        insert into tmp select name from student where mark < :mark;
        return 'Success: flag is 0';
    else
        return 'Unexpected flag input.';
    end if;

end;
$$;
call example_if(1, 75);
select * from tmp;
call example_if(0, 75);
select * from tmp;

--

create or replace procedure example_if(flag integer, mark integer)
returns varchar
language sql
as
$$
begin
begin transaction;
    if (flag = 1) then
        insert into tmp select name from student where mark >= :mark;
        return 'Success: flag is 1';
    elseif (flag = 0) then
        insert into tmp select name from student where mark < :mark;
        return 'Success: flag is 0';
    else
        return 'Unexpected flag input.';
    end if;
commit ;

end;
$$;
call example_if(1, 75);



--
show columns in table tmp;

----------------
-- tasks
use role sysadmin;
create or replace task t1
    warehouse=compute_wh
    schedule='5 minutes'
    as select 'hello from t1';
execute task t1;
use role accountadmin;
grant execute task on account to role sysadmin;
use role sysadmin;
execute task t1;

use role accountadmin;
grant execute managed task on account to role sysadmin;
use role sysadmin;
create or replace task t1
    USER_TASK_MANAGED_INITIAL_WAREHOUSE_SIZE = 'XSMALL'
    schedule='5 minutes'
    as select 'hello from t1';
use role sysadmin;
execute task t1;


use role sysadmin;
create or replace task t2
    USER_TASK_MANAGED_INITIAL_WAREHOUSE_SIZE = 'XSMALL'
    after t1
    as select 'hello from t2';

use role sysadmin;
create or replace task t3
    USER_TASK_MANAGED_INITIAL_WAREHOUSE_SIZE = 'XSMALL'
    after t2,t1
    as select 'hello from t3';

describe task t3;

alter task t2 resume;
alter task t3 resume;

execute task t1;
execute task t2;

--DAG 2:
create or replace task task1
    USER_TASK_MANAGED_INITIAL_WAREHOUSE_SIZE = 'XSMALL'
    schedule='5 minutes'
    as select 'hello from task1';

create or replace task task2
    USER_TASK_MANAGED_INITIAL_WAREHOUSE_SIZE = 'XSMALL'
    after task1
    as select 'hello from task2';

create or replace task task3
    USER_TASK_MANAGED_INITIAL_WAREHOUSE_SIZE = 'XSMALL'
    schedule='5 minutes'
    as select 'hello from task3';

create or replace task task4
    USER_TASK_MANAGED_INITIAL_WAREHOUSE_SIZE = 'XSMALL'
    after task2,task3
    as select 'hello from task4';
```

```
---
create or replace task my_root_task
   USER_TASK_MANAGED_INITIAL_WAREHOUSE_SIZE = 'XSMALL'
   schedule='5 minutes'
   as select 'hello from my root task';


alter task task4 add after task2,task3, my_root_task;
-----------------



use role sysadmin;
create stage an_s3_bucket_stage
 url = 's3://uni-lab-files';
list @an_s3_bucket_stage;

 create or replace table soil_type_tbl
( plant_name varchar(25),
  soil_type number(1,0)
);

select $1 from @an_s3_bucket_stage/LU_SOIL_TYPE.tsv (file_format=>'')

copy into soil_type_tbl
from @an_s3_bucket
files = ( 'VEG_NAME_TO_SOIL_TYPE_PIPE.txt')
file_format = ( format_name=PIPECOLSEP_ONEHEADROW );

----
create or replace stage UNI_KLAUS_ZMD
  url = 's3://uni-klaus/zenas_metadata';
list @UNI_KLAUS_ZMD;

create or replace file format zmd_file_format_1
RECORD_DELIMITER = '^' field_delimiter = '=';

select $1,$2
from @uni_klaus_zmd/product_coordination_suggestions.txt
(file_format => zmd_file_format_1);

create or replace table mytbl (color varchar);

copy into mytbl from
   (select $1
from @uni_klaus_zmd/product_coordination_suggestions.txt
(file_format => zmd_file_format_1));

select * from mytbl;
truncate table mytbl;

create or replace procedure insert_file(file_name varchar)
returns varchar
language sql
as
$$
begin

execute immediate 'copy into mytbl from
   (select $1
from @uni_klaus_zmd/' || :file_name || '
(file_format => zmd_file_format_1))';

end;

$$;
call insert_file('product_coordination_suggestions.txt');

copy into mytbl from
   (select $1
from @uni_klaus_zmd/:file_name
(file_format => zmd_file_format_1));
-----------
select current_account();

create or replace stage json_stage;
list @json_stage;
create or replace file format json_file_format
   type='json'
   strip_outer_array = true;

select $1 from @json_stage (file_format => json_file_format);
list @json_stage;
```

```
select to_date($1:sale_date::text, 'MM-DD-YY') from @json_stage (file_format => json_file_format);

select to_date('2012-07-23', 'YYYY-MM-DD');
select to_date('2012-23-07', 'YYYY-DD-MM');
select to_date('12-07-23', 'YY-MM-DD');
select to_date('4-25-16', 'MM-DD-YY');

select $1:location.city::varchar,
 $1:location.zip::varchar,
 $1:sq__ft::number,
 $1:price::number,
to_date($1:sale_date::text, 'MM-DD-YY') from @json_stage (file_format => json_file_format);

---------------------

select $1:year_published::date from @json_stage/book.json (file_format => json_file_format);
select to_date($1:year_published::text, 'YYYY')
    from @json_stage/book.json (file_format => json_file_format);

select to_date('2001', 'YYYY');
select YEAR(to_date('2001-01-01', 'YYYY-MM-DD'));

SELECT YEAR(CURRENT_DATE())||'-'||WEEK(CURRENT_DATE());
SELECT YEAR(CURRENT_DATE());

select to_date($1:year_published::text) from @json_stage/book2.json (file_format => json_file_format);
select $1:year_published::date from @json_stage/book2.json (file_format => json_file_format);

select $1:year_published::date from @json_stage/book3.json (file_format => json_file_format);

select to_date($1:year_published::text, 'MM-DD-YY') from @json_stage/book3.json (file_format => json_file_format);

select to_date($1:year_published::text, 'MM-DD-YY')
    from @json_stage/book3.json (file_format => json_file_format);

----
create or replace table export_tbl (name text, age int);
insert into export_tbl values
    ('Tom', 33),
    ('John',29);
create or replace stage my_internal_stage;
copy into @my_internal_stage/export.tsv
    from (select * from export_tbl)
    file_format=(type='csv' field_delimiter='\t'COMPRESSION =none)
    overwrite=true;

list @my_internal_stage;
remove @my_internal_stage/export.tsv_0_0_0.csv.gz;

use role securityadmin;
create role query_role;
use role sysadmin;
select * from fengdb.information_schema.databases;
grant usage on database fengdb to role query_role;
grant usage on schema fengdb.information_schema to role query_role;
grant select on view fengdb.information_schema.databases to role query_role;

use role accountadmin;
show grants on schema fengdb.information_schema;
show grants to schema fengdb.information_schema;

create or replace view export_view as
    select * from export_tbl;
show views;
create or replace view export_view2 as
    select * from export_tbl;
show views;
select * from export_view;
--
create or replace stream export_stream on table export_tbl;
create or replace stream invoice_stream on table invoices;
create or replace stream soil_stream on table soil_type;
create or replace stream date_stream on table some_date;
create or replace stream student_stream on table student;

SHOW STREAMS IN SCHEMA fengdb.fengschema;
--SELECT ''SELECT '''''' || "name" || '''''' as stream_name, COUNT(*) cnt FROM '' || "fengdb" || ''.'' || "fengschema" || ''.'' || "name" CMD
--          FROM TABLE(RESULT_SCAN(LAST_QUERY_ID()));

select "name" as stream_name from TABLE(RESULT_SCAN(LAST_QUERY_ID()));

SELECT "SELECT """ || "name" || """ as stream_name, COUNT(*) cnt FROM " || "database_name" || "." || "schema_name" || "." || "name" CMD
         FROM TABLE(RESULT_SCAN(LAST_QUERY_ID()))
```

```
EXECUTE IMMEDIATE
$$;



----
select grader(step, (actual = expected), actual, expected, description) as graded_results from
(
SELECT
 'SMEW08' as step
 ,(select count(*)/NULLIF(count(*),0) from snowflake.reader_account_usage.query_history
where USER_NAME = 'MANAGED_READER_ADMIN' and query_text ilike ('%366%')) as actual
 , 1 as expected
 ,'03-00-01-08' as description
UNION ALL
SELECT
  'SMEW09' as step
 ,(select count(*)/NULLIF(count(*),0) from snowflake.reader_account_usage.query_history
where USER_NAME = 'MANAGED_READER_ADMIN' and query_text ilike ('%NCIES%')) as actual
 , 1 as expected
 ,'03-00-01-09' as description
UNION ALL
SELECT
  'SMEW10' as step
 ,(select count(*)/NULLIF(count(*),0) from snowflake.reader_account_usage.query_history
where USER_NAME = 'MANAGED_READER_ADMIN' and query_text ilike ('%IMPLE%')) as actual
 , 1 as expected
 ,'03-00-01-10' as description
UNION ALL
SELECT
    'SMEW11' as step
,(select count(*)/NULLIF(count(*),0) from snowflake.reader_account_usage.query_history
where USER_NAME = 'MANAGED_READER_ADMIN' and query_text ilike ('%DE_TO%')) as actual
, 1 as expected
,'03-00-01-11' as description
);



    ------------------------
DECLARE
    SQL VARCHAR;
    RS RESULTSET;

BEGIN
    SHOW STREAMS IN SCHEMA fengdb.fengschema;
    SQL := '
    WITH commands AS
    (
        SELECT ''SELECT '''''' || "name" || '''''' as stream_name, COUNT(*) cnt FROM '' || "database_name" || ''.'' || "schema_name" || ''.'' || "name" CMD
            FROM TABLE(RESULT_SCAN(LAST_QUERY_ID()))
    )
    SELECT listagg(cmd, '' UNION ALL \n'') FROM COMMANDS;

    ';

    RS := (EXECUTE IMMEDIATE :SQL);
    RETURN TABLE(RS);
END;
$$;

create or replace table order_tbl (amount float, order_date date);
insert into order_tbl values
    (12.5, to_date('25/05/2022','dd/mm/yyyy'));
select * from order_tbl;

create or replace table order_tbl (amount float, order_date text);
insert into order_tbl values
    (12.5, '25/05/2022'),
    (17.75, '27/05/2022'),
    (2.5, '12/06/2022'),
    (7.25, '17/06/2022'),
    (12.5, '25/06/2022'),
    (27.5, '02/07/2022'),
    (32.5, '15/07/2022'),
    (7.25, '20/07/2022')
    ;
--truncate table order_tbl;
select * from order_tbl;

select avg(amount), date_trunc(month, to_date(order_date,'dd/mm/yyyy')) as order_month from order_tbl
    group by order_month order by order_month;
```

```
--------------------

select grader(step, (actual = expected), actual, expected, description) as graded_results from
(
SELECT
 'SMEW08' as step
 ,(select count(*)/NULLIF(count(*),0) from snowflake.reader_account_usage.query_history
where USER_NAME = 'MANAGED_READER_ADMIN' and query_text ilike ('%366%')) as actual
 , 1 as expected
 ,'03-00-01-08' as description
UNION ALL
SELECT
  'SMEW09' as step
 ,(select count(*)/NULLIF(count(*),0) from snowflake.reader_account_usage.query_history
where USER_NAME = 'MANAGED_READER_ADMIN' and query_text ilike ('%NCIES%')) as actual
 , 1 as expected
 ,'03-00-01-09' as description
UNION ALL
SELECT
  'SMEW10' as step
 ,(select count(*)/NULLIF(count(*),0) from snowflake.reader_account_usage.query_history
where USER_NAME = 'MANAGED_READER_ADMIN' and query_text ilike ('%IMPLE%')) as actual
 , 1 as expected
 ,'03-00-01-10' as description
UNION ALL
SELECT
    'SMEW11' as step
,(select count(*)/NULLIF(count(*),0) from snowflake.reader_account_usage.query_history
where USER_NAME = 'MANAGED_READER_ADMIN' and query_text ilike ('%DE_TO%')) as actual
, 1 as expected
,'03-00-01-11' as description
);
----

    ----
create or replace
---- procedure in python
CREATE OR REPLACE PROCEDURE MYPROC(from_table STRING, to_table STRING, count INT)
  RETURNS STRING
  LANGUAGE PYTHON
  RUNTIME_VERSION = '3.8'
  PACKAGES = ('snowflake-snowpark-python')
  HANDLER = 'run'
AS
$$
def run(session, from_table, to_table, count):
  session.table(from_table).limit(count).write.save_as_table(to_table)
  return "SUCCESS"
$$;
call MYPROC('SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.ORDERS', 'fengdb.fengschema.small_order_tbl', 5);
select * from fengdb.fengschema.small_order_tbl;

select * from information_schema.packages where package_name = 'snowflake-snowpark-python' order by version desc;

select distinct language from information_schema.packages;
select distinct package_name from information_schema.packages where package_name like '%snowpark%';

select count(distinct package_name) from information_schema.packages where language='python';
select * from information_schema.packages where language='python' and package_name like'%numpy%';
select * from information_schema.packages where language='python' and package_name like'%pandas%';
select * from information_schema.packages where language='python' and package_name like'%pytorch%';

execute immediate 'session.table(SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.ORDERS).limit(10).write.save_as_table(fengdb.fengschema.small_order_tbl)';

$$
def run(session, from_table, to_table, count):
  session.table(from_table).limit(count).write.save_as_table(to_table)
  return "SUCCESS"
$$;

----
CREATE OR REPLACE PROCEDURE MYPROC(from_table STRING, to_table STRING, count INT)
  RETURNS STRING
  LANGUAGE PYTHON
  RUNTIME_VERSION = '3.8'
  PACKAGES = ('snowflake-snowpark-python')
  HANDLER = 'run'
AS
$$
import numpy as np
my_array = np.array([1,2,3,4])
```

```
def run(session, from_table, to_table, count):
  session.table(from_table).limit(count).write.save_as_table(to_table)
  return "SUCCESS"
$$;
call MYPROC('SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.ORDERS', 'fengdb.fengschema.small_order_tbl2', 5);


----
list @fengdb.fengschema.my_internal_stage;
CREATE OR REPLACE PROCEDURE MYPROC2(from_table STRING, to_table STRING, count INT)
  RETURNS STRING
  LANGUAGE PYTHON
  RUNTIME_VERSION = '3.8'
  PACKAGES = ('snowflake-snowpark-python')
  IMPORTS = ('@fengdb.fengschema.my_internal_stage/myproc.py')
  HANDLER = 'myproc.run';
call MYPROC2('SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.ORDERS', 'fengdb.fengschema.small_order_tbl3', 5);
select * from fengdb.fengschema.small_order_tbl3;

----
CREATE OR REPLACE PROCEDURE MYPROC_SQL(from_table STRING, to_table STRING, count INT)
  RETURNS STRING
  LANGUAGE SQL
  as
  $$
 begin
    create or replace table fengdb.fengschema.small_order_tbl5 as select * from SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.ORDERS limit :count;
    return 'Success';
end;
  $$;
call MYPROC_SQL('SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.ORDERS', 'fengdb.fengschema.small_order_tbl5', 5);

select * from fengdb.fengschema.small_order_tbl5;

create or replace table fengdb.fengschema.small_order_tbl5 as
    (select * from SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.ORDERS limit 10);

execute immediate $$
begin
    create or replace table fengdb.fengschema.small_order_tbl5 as select * from SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.ORDERS limit 5;
    return 'Success';
end;
$$;

    execute immediate $$
declare
  target_date date;
  mydates cursor for select mydate from some_date;
begin
  open mydates;
  for myrow in mydates do
     target_date := myrow.mydate::date;
     insert into new_data_tbl select price, mydate from data_tbl where mydate = :target_date;
  end for;
  close mydates;
  return 'success';
end;
$$;

-----
list @my_internal_stage;

create or replace procedure myprocedure(tablename text)
    RETURNS STRING
    LANGUAGE PYTHON
    RUNTIME_VERSION = '3.8'
    PACKAGES = ('snowflake-snowpark-python==1.0.0')
    IMPORTS=('@my_internal_stage/mysnowpark.py')
    HANDLER = 'mysnowpark.run';

 -- mysnowpark.py
from snowflake.snowpark.types import StringType
def run(session,table_name):
    df = session.table(table_name)
    mod5_udf = session.udf.register(
        alpha_chk,
        return_type=StringType(),
        input_types=[StringType()],
    )
    df.select(mod5_udf('EMPNAME'));

----
select current_account();
```

```
----

    use role SYSADMIN;
--Caden set up a new database (and you will, too)
create database ACME;

--did Snowflake set your worksheet database to the new database?
--if not, you can do it yourself.
use database ACME;

--get rid of the public schema - too generic
drop schema PUBLIC;

--When creating shares it is best to have multiple schemas
create schema ACME.SALES;
create schema ACME.STOCK;
create schema ACME.ADU; --this is the schema they'll use to share to ADU, Max's company

use role SYSADMIN;
--Lottie's team will enter new stock into this table when inventory is received
-- the Date_Sold and Customer_Id will be null until the car is sold
create or replace table ACME.STOCK.LOTSTOCK
(
 VIN VARCHAR(17)
,EXTERIOR VARCHAR(50)
,INTERIOR VARCHAR(50)
,DATE_SOLD DATE
,CUSTOMER_ID NUMBER(20)
);

--This secure view breaks the VIN into digestible components
--this view only shares unsold cars because the unsold cars
--are the ones that need to be enhanced
create or replace secure view ACME.ADU.LOTSTOCK
AS (
SELECT VIN
  , LEFT(VIN,3) as WMI
  , SUBSTR(VIN,4,5) as VDS
  , SUBSTR(VIN,10,1) as MODYEARCODE
  , SUBSTR(VIN,11,1) as PLANTCODE
  , EXTERIOR
  , INTERIOR
FROM ACME.STOCK.LOTSTOCK
WHERE DATE_SOLD is NULL
);

--You need a file format if you want to load the table
create file format ACME.STOCK.COMMA_SEP_HEADERROW
TYPE = 'CSV'
COMPRESSION = 'AUTO'
FIELD_DELIMITER = ','
RECORD_DELIMITER = '\n'
SKIP_HEADER = 1
FIELD_OPTIONALLY_ENCLOSED_BY = '\042'
TRIM_SPACE = TRUE
ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE
ESCAPE = 'NONE'
ESCAPE_UNENCLOSED_FIELD = '\134'
DATE_FORMAT = 'AUTO'
TIMESTAMP_FORMAT = 'AUTO'
NULL_IF = ('\\N');

--Use a COPY INTO to load the data
--the file is named Lotties_LotStock_Data.csv

COPY INTO acme.stock.lotstock
from @intl_db.public.like_a_window_into_an_s3_bucket
files = ('smew/Lotties_LotStock_Data.csv')
file_format =(format_name=ACME.STOCK.COMMA_SEP_HEADERROW);


-- After loading your base table is no longer empty
-- it should now have 300 rows
select * from acme.stock.lotstock;

--the View will show just 298 rows because the view only shows
--rows where the date_sold is null
select * from acme.adu.lotstock;


    -- set your worksheet drop lists to the location of your GRADER function
--DO NOT EDIT ANYTHING BELOW THIS LINE
```

```
select grader(step, (actual = expected), actual, expected, description) as graded_results from (
 SELECT 'SMEW12' as step
 ,(select count(*)
   from SNOWFLAKE.ACCOUNT_USAGE.DATABASES
   where database_name in ('INTL_DB','DEMO_DB','ACME', 'ACME_DETROIT','ADU_VINHANCED')
   and deleted is null) as actual
 , 5 as expected
 ,'Databases from all over!' as description
);

use role accountadmin;
alter database acme_vinventory rename to acme_detroit;
alter database vinhance_shareback rename to adu_vinhanced;
show databases;
select count(*)
   from SNOWFLAKE.ACCOUNT_USAGE.DATABASES
   where database_name in ('INTL_DB','DEMO_DB','ACME', 'ACME_DETROIT','ADU_VINHANCED')
   and deleted is null;

select * from SNOWFLAKE.ACCOUNT_USAGE.DATABASES;



USE ROLE SYSADMIN;

--Max created a database to store Vehicle Identification Numbers
CREATE DATABASE max_vin;

DROP SCHEMA max_vin.public;
CREATE SCHEMA max_vin.decode;

--We need a table that will allow WMIs to be decoded into Manufacturer Name, Country and Vehicle Type
CREATE TABLE MAX_VIN.DECODE.WMITOMANUF
(
     WMI         VARCHAR(6)
    ,MANUF_ID       NUMBER(6)
    ,MANUF_NAME             VARCHAR(50)
    ,COUNTRY         VARCHAR(50)
    ,VEHICLETYPE    VARCHAR(50)
 );

--We need a table that will allow you to go from Manufacturer to Make
--For example, Mercedes AG of Germany and Mercedes USA both roll up into Mercedes
--But they use different WMI Codes
CREATE TABLE MAX_VIN.DECODE.MANUFTOMAKE
(
     MANUF_ID    NUMBER(6)
    ,MAKE_NAME             VARCHAR(50)
    ,MAKE_ID       NUMBER(5)
);

--We need a table that can decode the model year
-- The year 2001 is represented by the digit 1
-- The year 2020 is represented by the letter L
CREATE TABLE MAX_VIN.DECODE.MODELYEAR
(
     MODYEARCODE         VARCHAR(1)
    ,MODYEARNAME        NUMBER(4)
);

--We need a table that can decode which plant at which
--the vehicle was assembled
--You might have code "A" for Honda and code "A" for Ford
--so you need both the Make and the Plant Code to properly decode
--the plant code
CREATE TABLE MAX_VIN.DECODE.MANUFPLANTS
(
     MAKE_ID       NUMBER(5)
    ,PLANTCODE  VARCHAR(1)
    ,PLANTNAME VARCHAR(75)
 );

--We need to use a combination of both the Make and VDS
--to decode many attributes including the engine, transmission, etc
CREATE TABLE MAX_VIN.DECODE.MMVDS
(
     MAKE_ID       NUMBER(3)
    ,MODEL_ID    NUMBER(6)
    ,MODEL_NAME          VARCHAR(50)
    ,VDS  VARCHAR(5)
    ,DESC1           VARCHAR(25)
    ,DESC2           VARCHAR(25)
    ,DESC3           VARCHAR(50)
    ,DESC4          VARCHAR(25)
    ,DESC5          VARCHAR(25)
    ,BODYSTYLE   VARCHAR(25)
    ,ENGINE         VARCHAR(100)
    ,DRIVETYPE   VARCHAR(50)
    ,TRANS          VARCHAR(50)
    ,MPG VARCHAR(25)
);


    --Create a file format and then load each of the 5 Lookup Tables
--You need a file format if you want to load the table
CREATE FILE FORMAT MAX_VIN.DECODE.COMMA_SEP_HEADERROW
TYPE = 'CSV'
COMPRESSION = 'AUTO'
FIELD_DELIMITER = ','
RECORD_DELIMITER = '\n'
SKIP_HEADER = 1
FIELD_OPTIONALLY_ENCLOSED_BY = '\042'
TRIM_SPACE = TRUE
ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE
ESCAPE = 'NONE'
ESCAPE_UNENCLOSED_FIELD = '\134'
DATE_FORMAT = 'AUTO'
TIMESTAMP_FORMAT = 'AUTO'
NULL_IF = ('\\N');

-----------------------
list @intl_db.public.like_a_window_into_an_s3_bucket/smew;
/*
smew/Maxs_MMVDS_Data.csv
smew/Maxs_ManufPlants_Data.csv
smew/Maxs_ManufToMake_Data.csv
smew/Maxs_ModelYear_Data.csv
smew/Maxs_WMIToManuf_data.csv
*/

COPY INTO MAX_VIN.DECODE.WMITOMANUF
from @intl_db.public.like_a_window_into_an_s3_bucket
files = ('smew/Maxs_WMIToManuf_data.csv')
file_format =(format_name=MAX_VIN.DECODE.COMMA_SEP_HEADERROW);

COPY INTO MAX_VIN.DECODE.MANUFTOMAKE
from @intl_db.public.like_a_window_into_an_s3_bucket
files = ('smew/Maxs_ManufToMake_Data.csv')
file_format =(format_name=MAX_VIN.DECODE.COMMA_SEP_HEADERROW);

COPY INTO MAX_VIN.DECODE.MODELYEAR
from @intl_db.public.like_a_window_into_an_s3_bucket
files = ('smew/Maxs_ModelYear_Data.csv')
file_format =(format_name=MAX_VIN.DECODE.COMMA_SEP_HEADERROW);

COPY INTO MAX_VIN.DECODE.MANUFPLANTS
from @intl_db.public.like_a_window_into_an_s3_bucket
files = ('smew/Maxs_ManufPlants_Data.csv')
file_format =(format_name=MAX_VIN.DECODE.COMMA_SEP_HEADERROW);

COPY INTO MAX_VIN.DECODE.MMVDS
from @intl_db.public.like_a_window_into_an_s3_bucket
files = ('smew/Maxs_MMVDS_Data.csv')
file_format =(format_name=MAX_VIN.DECODE.COMMA_SEP_HEADERROW);


    ------------------------
--Max has Lottie's VINventory table. Now he'll join his decode tables to the data
-- He'll create a select statement that ties each table into Lottie's VINS
-- Every time he adds a new table, he'll make sure he still has 298 rows

SELECT *
FROM ACME_DETROIT.ADU.LOTSTOCK l-- he uses Lottie's data from the INBOUND SHARE
JOIN MAX_VIN.DECODE.MODELYEAR y -- and confirms he can join it with his own decode data
ON l.modyearcode=y.modyearcode;

SELECT *
FROM ACME_DETROIT.ADU.LOTSTOCK l -- he uses Lottie's data from the INBOUND SHARE
JOIN MAX_VIN.DECODE.WMITOMANUF w -- and confirms he can join it with his own decode data
ON l.WMI=w.WMI;

--Add the next table (still 298?)
SELECT *
FROM ACME_DETROIT.ADU.LOTSTOCK l -- he uses Lottie's data from the INBOUND SHARE
JOIN MAX_VIN.DECODE.WMITOMANUF w -- and confirms he can join it with his own decode data
ON l.WMI=w.WMI
```

```
JOIN MAX_VIN.DECODE.MANUFTOMAKE m
ON w.manuf_id=m.manuf_id;

--Until finally he has all 5 lookup tables added
--He can then remove the asterisk and start narrowing down the
--fields to include in the final output
SELECT
l.VIN
,y.MODYEARNAME
,m.MAKE_NAME
,v.DESC1
,v.DESC2
,v.DESC3
,BODYSTYLE
,ENGINE
,DRIVETYPE
,TRANS
,MPG
,MANUF_NAME
,COUNTRY
,VEHICLETYPE
,PLANTNAME
FROM ACME_DETROIT.ADU.LOTSTOCK l -- he joins Lottie's data from the INBOUND SHARE
JOIN MAX_VIN.DECODE.WMITOMANUF w -- with all his data (he just tested)
   ON l.WMI=w.WMI
JOIN MAX_VIN.DECODE.MANUFTOMAKE m
   ON w.manuf_id=m.manuf_id
JOIN MAX_VIN.DECODE.MANUFPLANTS p
   ON l.plantcode=p.plantcode
   AND m.make_id=p.make_id
JOIN MAX_VIN.DECODE.MMVDS v
   ON v.vds=l.vds
   and v.make_id = m.make_id
JOIN MAX_VIN.DECODE.MODELYEAR y
   ON l.modyearcode=y.modyearcode;


-- Once the select statement looks good (above), Max lays a view on top of it
-- this will make it easier to use in a Stored procedure

USE ROLE SYSADMIN;
CREATE DATABASE MAX_OUTGOING; --this new database will be used for his OUTBOUND SHARE
CREATE SCHEMA MAX_OUTGOING.FOR_ACME; --this schema he creates especially for ACME

-- This is a live view of the data Lottie and Caden Need!
CREATE OR REPLACE SECURE VIEW MAX_OUTGOING.FOR_ACME.LOTSTOCKENHANCED as
(
SELECT
l.VIN
,y.MODYEARNAME
,m.MAKE_NAME
,v.DESC1
,v.DESC2
,v.DESC3
,BODYSTYLE
,ENGINE
,DRIVETYPE
,TRANS
,MPG
,EXTERIOR
,INTERIOR
,MANUF_NAME
,COUNTRY
,VEHICLETYPE
,PLANTNAME
FROM ACME_DETROIT.ADU.LOTSTOCK l
JOIN MAX_VIN.DECODE.WMITOMANUF w
   ON l.WMI=w.WMI
JOIN MAX_VIN.DECODE.MANUFTOMAKE m
   ON w.manuf_id=m.manuf_id
JOIN MAX_VIN.DECODE.MANUFPLANTS p
   ON l.plantcode=p.plantcode
   AND m.make_id=p.make_id
JOIN MAX_VIN.DECODE.MMVDS v
   ON v.vds=l.vds and v.make_id = m.make_id
JOIN MAX_VIN.DECODE.MODELYEAR y
   ON l.modyearcode=y.modyearcode
);

  ------
    -- Even though it would be nice to share the view back to Lottie,
-- You can't share a share so we have to make a copy of the data to share back
```

```
CREATE OR REPLACE TABLE MAX_OUTGOING.FOR_ACME.LOTSTOCKRETURN
(
     VIN          VARCHAR(17)
    ,MODYEARNAME       NUMBER(4)
    ,MAKE_NAME              VARCHAR(50)
    ,DESC1               VARCHAR(50)
    ,DESC2               VARCHAR(50)
    ,DESC3               VARCHAR(50)
    ,BODYSTYLE      VARCHAR(25)
    ,ENGINE              VARCHAR(100)
    ,DRIVETYPE      VARCHAR(50)
    ,TRANS               VARCHAR(50)
    ,MPG       VARCHAR(25)
    ,EXTERIOR       VARCHAR(50)
    ,INTERIOR       VARCHAR(50)
    ,MANUF_NAME            VARCHAR(50)
    ,COUNTRY        VARCHAR(50)
    ,VEHICLETYPEVARCHAR(50)
    ,PLANTNAME    VARCHAR(75)
  );



-------------------------------------------------------------------------------
    USE ROLE SYSADMIN;

create or replace procedure lotstockupdate_sp()
  returns string not null
  language javascript
  as
  $$
    var my_sql_command1 = "truncate table max_outgoing.for_acme.lotstockreturn;";
    var statement1 = snowflake.createStatement( {sqlText: my_sql_command1} );
    var result_set1 = statement1.execute();

    var my_sql_command2 ="insert into max_outgoing.for_acme.lotstockreturn ";
    my_sql_command2 += "select VIN, MODYEARNAME, MAKE_NAME, DESC1, DESC2, DESC3, BODYSTYLE";
    my_sql_command2 += ",ENGINE, DRIVETYPE, TRANS, MPG, EXTERIOR, INTERIOR, MANUF_NAME, COUNTRY, VEHICLETYPE, PLANTNAME";
    my_sql_command2 += " from max_outgoing.for_acme.lotstockenhanced;";

    var statement2 = snowflake.createStatement( {sqlText: my_sql_command2} );
    var result_set2 = statement2.execute();
    return my_sql_command2;
   $$;

   --View your Stored Procedure
   show procedures in account;
   desc procedure lotstockupdate_sp();


    --==========SCHEDULED TASK=============================================
-- Create a task that calls the stored procedure every hour
-- so that Lottie sees updates at least every hour

USE ROLE ACCOUNTADMIN;
grant execute task on account to role sysadmin;

USE ROLE SYSADMIN;
create or replace task acme_return_update
  warehouse = compute_wh
  schedule = '1 minute'
as
  call lotstockupdate_sp();

--if you need to see who owns the task
show grants on task acme_return_update;

--Look at the task you just created to make sure it turned out okay
show tasks;
desc task acme_return_update;

--if you task has a state of "suspended" run this to get it going
alter task acme_return_update resume;

--Check back 5 mins later to make sure your task has been running
--You will not be able to see your task on the Query History Tab
select *
  from table(information_schema.task_history())
  order by scheduled_time;


--=== CHECK ON AND SUSPEND THE SCHEDULED TASK =========================
show tasks in account;
```

```
desc task acme_return_update;

alter task acme_return_update suspend;

--Check back 5 mins later to make sure your task is NOT running
desc task acme_return_update;


select grader(step, (actual = expected), actual, expected, description) as graded_results from
(
SELECT 'SMEW13' as step
 ,(select count(*)
   from MAX_OUTGOING.FOR_ACME.LOTSTOCKRETURN) as actual
 , 298 as expected
 ,'LotStockReturn Table loaded' as description
);


-- set the worksheet drop lists to match the location of your GRADER function
--DO NOT MAKE ANY CHANGES BELOW THIS LINE
select grader(step, (actual = expected), actual, expected, description) as graded_results from (
SELECT
  'SMEW14' as step
 ,(select count(*)
   from DEMO_DB.PUBLIC.DETROIT_ZIPS) as actual
 , 9 as expected
 ,'Detroit Zips' as description
);

    ---------------
 --  2022-11-29T15:49:31.000+0000
 --   2018-12-12T07:27:29.000+0000
  select to_timestamp('2018-12-12T07:27:29.000+0000', 'yyyy-mm-ddThh:MM:ss');

  select to_timestamp('2018-12-12T07:27:29.000+0000');

select to_timestamp(replace(replace('2018-12-12T07:27:29.000+0000','T', ' '), '.000+0000'));


    select current_account();

select SHA2(IFNULL('', ''));

create or replace table mytable2 (totalamt varchar(16777216));
insert into mytable2 values ('0.27');
select cast(TOTALAMT as number(38,5)) AS TOTALAMT_NUM from mytable2;

select TO_NUMBER(TOTALAMT,38,5) AS TOTALAMT_NUM from mytable2;

----
create or replace table vartab (n number(2), v variant);

insert into vartab
    select column1 as n, parse_json(column2) as v
    from values (1, 'null'),
                (2, null),
                (3, 'true'),
                (4, '-17'),
                (5, '123.12'),
                (6, '1.912e2'),
                (7, '"Om ara pa ca na dhih"  '),
                (8, '[-1, 12, 289, 2188, false,]'),
                (9, '{ "x" : "abc", "y" : false, "z": 10} ')
       as vals;
select n, v, typeof(v)
    from vartab
    order by n;

-- parse_json means to parse an object assuming it's json.
Select parse_json(v):x::string as personinfo from vartab where n=9;

show users;
alter user scouterfeng set rsa_public_key='MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAypxhRYI4Ze3UyWr0g4j/
...
7YEYU2JBTc3a51aSfNFjvmWKwdWgWLcZEGectOZN3ZnOovCveXe4NDthb2TYnltm
KQIDAQAB';


create or replace table TEST(CODE string, LABEL string);
select * from test;
truncate table test;
```

```
-------------------------------------------------------------
alter database sf_sample_data rename to snowflake_sample_data;

-- on a shared database, imported privileges are pre-defined
-- by the owner of the data source for maximum security
GRANT IMPORTED PRIVILEGES
ON DATABASE SNOWFLAKE_SAMPLE_DATA
TO ROLE SYSADMIN;


--Check the range of values in the Market Segment Column
SELECT DISTINCT c_mktsegment
FROM SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.CUSTOMER;

--Find out which Market Segments have the most customers
SELECT c_mktsegment, COUNT(*)
FROM SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.CUSTOMER
GROUP BY c_mktsegment
ORDER BY COUNT(*);

-- Nations Table
SELECT N_NATIONKEY, N_NAME, N_REGIONKEY
FROM SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.NATION;

-- Regions Table
SELECT R_REGIONKEY, R_NAME
FROM SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.REGION;

-- Join the Tables and Sort
SELECT R_NAME as Region, N_NAME as Nation
FROM SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.NATION
JOIN SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.REGION
ON N_REGIONKEY = R_REGIONKEY
ORDER BY R_NAME, N_NAME ASC;

--Group and Count Rows Per Region
SELECT R_NAME as Region, count(N_NAME) as NUM_COUNTRIES
FROM SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.NATION
JOIN SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.REGION
ON N_REGIONKEY = R_REGIONKEY
GROUP BY R_NAME;

-- setup Dora
use role accountadmin;
create or replace api integration dora_api_integration
api_provider = aws_api_gateway
api_aws_role_arn = 'arn:aws:iam::321463406630:role/snowflakeLearnerAssumedRole'
enabled = true
api_allowed_prefixes = ('https://awy6hshxy4.execute-api.us-west-2.amazonaws.com/dev/edu_dora');

show integrations;

 use role accountadmin;
create or replace external function demo_db.public.grader(
      step varchar
    , passed boolean
    , actual integer
    , expected integer
    , description varchar)
returns variant
api_integration = dora_api_integration
context_headers = (current_timestamp,current_account, current_statement)
as 'https://awy6hshxy4.execute-api.us-west-2.amazonaws.com/dev/edu_dora/grader'
;

-- where did you put the function?
show functions in account;

-- did you put it here?
select *
from snowflake.account_usage.functions
where function_name = 'GRADER'
and function_catalog = 'DEMO_DB'
and function_owner = 'ACCOUNTADMIN';

-- set your worksheet drop lists to the location of your GRADER function

--DO NOT EDIT BELOW THIS LINE - Don't add or take away spaces, do change 'from' to 'FROM' - NO EDITS AT ALL
select GRADER(step,(actual = expected), actual, expected, description) as graded_results from (
SELECT 'DORA_IS_WORKING' as step
 ,(select 223 ) as actual
 ,223 as expected
 ,'Dora is working!' as description
);

-- lesson 3
USE ROLE SYSADMIN;
CREATE DATABASE INTL_DB;
USE SCHEMA INTL_DB.PUBLIC;

CREATE WAREHOUSE INTL_WH
WITH WAREHOUSE_SIZE = 'XSMALL'
WAREHOUSE_TYPE = 'STANDARD'
AUTO_SUSPEND = 600
AUTO_RESUME = TRUE;

USE WAREHOUSE INTL_WH;

use role accountadmin;
-- set your worksheet drop lists to the location of your GRADER function using commands
-- change the next two lines (if needed) to the location of your GRADER function
use database demo_db;
use schema public;

--DO NOT EDIT BELOW THIS LINE
select grader(step, (actual = expected), actual, expected, description) as graded_results from(
 SELECT 'SMEW01' as step
 ,(select count(*)
   from snowflake.account_usage.databases
   where database_name = 'INTL_DB'
   and deleted is null) as actual
 , 1 as expected
 ,'Created INTL_DB' as description
 );

use role sysadmin;
CREATE OR REPLACE TABLE INTL_DB.PUBLIC.INT_STDS_ORG_3661
(ISO_COUNTRY_NAME varchar(100),
 COUNTRY_NAME_OFFICIAL varchar(200),
 SOVEREIGNTY varchar(40),
 ALPHA_CODE_2DIGIT varchar(2),
 ALPHA_CODE_3DIGIT varchar(3),
 NUMERIC_COUNTRY_CODE integer,
 ISO_SUBDIVISION varchar(15),
 INTERNET_DOMAIN_CODE varchar(10)
);

--drop table INTL_DB.PUBLIC.INT_STDS_ORG_3661;
CREATE OR REPLACE FILE FORMAT INTL_DB.PUBLIC.PIPE_DBLQUOTE_HEADER_CR
  TYPE = 'CSV'
  COMPRESSION = 'AUTO'
  FIELD_DELIMITER = '|'
  RECORD_DELIMITER = '\r'
  SKIP_HEADER = 1
  FIELD_OPTIONALLY_ENCLOSED_BY = '\042'
  TRIM_SPACE = FALSE
  ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE
  ESCAPE = 'NONE'
  ESCAPE_UNENCLOSED_FIELD = '\134'
  DATE_FORMAT = 'AUTO'
  TIMESTAMP_FORMAT = 'AUTO'
  NULL_IF = ('\\N');

create stage like_a_window_into_an_s3_bucket
    url = 's3://uni-lab-files';
list @like_a_window_into_an_s3_bucket;

copy into INTL_DB.PUBLIC.INT_STDS_ORG_3661
from @like_a_window_into_an_s3_bucket
files = ( 'smew/ISO_Countries_UTF8_pipe.csv')
file_format = ( format_name='INTL_DB.PUBLIC.PIPE_DBLQUOTE_HEADER_CR' );

SELECT count(*) as FOUND, '249' as EXPECTED
FROM INTL_DB.PUBLIC.INT_STDS_ORG_3661;

select count(*) as OBJECTS_FOUND
from INTL_DB.INFORMATION_SCHEMA.TABLES
where table_schema='PUBLIC'
and table_name= 'INT_STDS_ORG_3661';

select row_count
from INTL_DB.INFORMATION_SCHEMA.TABLES
where table_schema='PUBLIC'
and table_name= 'INT_STDS_ORG_3661';
```

```
use role accountadmin;

-- set your worksheet drop lists to the location of your GRADER function using commands
-- change the next two lines (if needed) to the location of your GRADER function
use database demo_db;
use schema public;

--DO NOT EDIT BELOW THIS LINE
select grader(step, (actual = expected), actual, expected, description) as graded_results from(
SELECT 'SMEW02' as step
 ,(select count(*)
   from INTL_DB.INFORMATION_SCHEMA.TABLES
   where table_schema = 'PUBLIC'
   and table_name = 'INT_STDS_ORG_3661') as actual
 , 1 as expected
 ,'ISO table created' as description
);

select grader(step, (actual = expected), actual, expected, description) as graded_results from(
SELECT 'SMEW03' as step
 ,(select row_count
   from INTL_DB.INFORMATION_SCHEMA.TABLES
   where table_name = 'INT_STDS_ORG_3661') as actual
 , 249 as expected
 ,'ISO Table Loaded' as description
);


CREATE VIEW NATIONS_SAMPLE_PLUS_ISO (iso_country_name, country_name_official,alpha_code_2digit, region) AS
SELECT
    iso_country_name
    , country_name_official,alpha_code_2digit
    ,r_name as region
FROM INTL_DB.PUBLIC.INT_STDS_ORG_3661 i
LEFT JOIN SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.NATION n
ON UPPER(i.iso_country_name)=n.n_name
LEFT JOIN SNOWFLAKE_SAMPLE_DATA.TPCH_SF1.REGION r
ON n_regionkey = r_regionkey;

SELECT *
FROM INTL_DB.PUBLIC.NATIONS_SAMPLE_PLUS_ISO;

select grader(step, (actual = expected), actual, expected, description) as graded_results from(
SELECT 'SMEW04' as step
 ,(select count(*)
   from INTL_DB.PUBLIC.NATIONS_SAMPLE_PLUS_ISO) as actual
 , 249 as expected
 ,'Nations Sample Plus Iso' as description
);


CREATE TABLE INTL_DB.PUBLIC.CURRENCIES
(
  CURRENCY_ID INTEGER,
  CURRENCY_CHAR_CODE varchar(3),
  CURRENCY_SYMBOL varchar(4),
  CURRENCY_DIGITAL_CODE varchar(3),
  CURRENCY_DIGITAL_NAME varchar(30)
)
  COMMENT = 'Information about currencies including character codes, symbols, digital codes, etc.';

CREATE TABLE INTL_DB.PUBLIC.COUNTRY_CODE_TO_CURRENCY_CODE
  (
    COUNTRY_CHAR_CODE Varchar(3),
    COUNTRY_NUMERIC_CODE INTEGER,
    COUNTRY_NAME Varchar(100),
    CURRENCY_NAME Varchar(100),
    CURRENCY_CHAR_CODE Varchar(3),
    CURRENCY_NUMERIC_CODE INTEGER
  )
  COMMENT = 'Many to many code lookup table';


  CREATE FILE FORMAT INTL_DB.PUBLIC.CSV_COMMA_LF_HEADER
  TYPE = 'CSV'
  COMPRESSION = 'AUTO'
  FIELD_DELIMITER = ','
  RECORD_DELIMITER = '\n'
  SKIP_HEADER = 1
  FIELD_OPTIONALLY_ENCLOSED_BY = 'NONE'
  TRIM_SPACE = FALSE
  ERROR_ON_COLUMN_COUNT_MISMATCH = TRUE
  ESCAPE = 'NONE'
  ESCAPE_UNENCLOSED_FIELD = '\134'
  DATE_FORMAT = 'AUTO'
  TIMESTAMP_FORMAT = 'AUTO'
  NULL_IF = ('\\N');

list @like_a_window_into_an_s3_bucket/smew;

copy into CURRENCIES
    from @like_a_window_into_an_s3_bucket
files = ( 'smew/currencies.csv')
file_format = ( format_name='INTL_DB.PUBLIC.CSV_COMMA_LF_HEADER' );

copy into COUNTRY_CODE_TO_CURRENCY_CODE
    from @like_a_window_into_an_s3_bucket
files = ( 'smew/country_code_to_currency_code.csv')
file_format = ( format_name='INTL_DB.PUBLIC.CSV_COMMA_LF_HEADER' );

select grader(step, (actual = expected), actual, expected, description) as graded_results from(
SELECT 'SMEW05' as step
 ,(select row_count
  from INTL_DB.INFORMATION_SCHEMA.TABLES
  where table_schema = 'PUBLIC'
  and table_name = 'COUNTRY_CODE_TO_CURRENCY_CODE') as actual
 , 265 as expected
 ,'CCTCC Table Loaded' as description
);
select grader(step, (actual = expected), actual, expected, description) as graded_results from(
SELECT 'SMEW06' as step
 ,(select row_count
  from INTL_DB.INFORMATION_SCHEMA.TABLES
  where table_schema = 'PUBLIC'
  and table_name = 'CURRENCIES') as actual
 , 151 as expected
 ,'Currencies table loaded' as description
);

create or replace view simple_currency (cty_code, cur_code) as
select country_char_code, currency_char_code from INTL_DB.PUBLIC.COUNTRY_CODE_TO_CURRENCY_CODE;

select * from simple_currency;

select grader(step, (actual = expected), actual, expected, description) as graded_results from(
 SELECT 'SMEW07' as step
,(select count(*)
  from INTL_DB.PUBLIC.SIMPLE_CURRENCY ) as actual
, 265 as expected
,'Simple Currency Looks Good' as description
);


-- lesson 4, outbind share
select current_account();

use role accountadmin;
grant override share restrictions on account to role accountadmin;

ALTER VIEW INTL_DB.PUBLIC.NATIONS_SAMPLE_PLUS_ISO
SET SECURE;

ALTER VIEW INTL_DB.PUBLIC.SIMPLE_CURRENCY
SET SECURE;

SELECT YEAR(CURRENT_DATE())||'-'||WEEK(CURRENT_DATE());

use role accountadmin;
SHOW MANAGED ACCOUNTS;
```