2025-07-11 Meeting notes

Your SQL script has a few syntax errors and inconsistencies. Let me correct and explain each part step by step for Amazon Redshift (or PostgreSQL-like syntax):

✅ Corrected script

- ```
  -- Create the user
  CREATE USER readonly_user PASSWORD 'your_password_here';

  -- Grant usage on the target schema
  GRANT USAGE ON SCHEMA commercialanalytics TO readonly_user;

  -- Grant select on all existing tables in the schema
  GRANT SELECT ON ALL TABLES IN SCHEMA commercialanalytics TO readonly_user;

  -- Ensure future tables also have SELECT for this user
  ALTER DEFAULT PRIVILEGES IN SCHEMA commercialanalytics
  GRANT SELECT ON TABLES TO readonly_user;

  -- Revoke usage on the public schema from this user
  REVOKE USAGE ON SCHEMA public FROM readonly_user;

  -- Revoke all privileges on tables in the public schema from this user
  REVOKE ALL PRIVILEGES ON ALL TABLES IN SCHEMA public FROM readonly_user;

  -- Optionally, revoke public access to public schema (be careful — affects all
  users)
  REVOKE USAGE ON SCHEMA public FROM PUBLIC;

  ```

⚠️ Issues in your original script

1. **Syntax errors in GRANT/REVOKE lines**:
   You wrote `readonly user` instead of `readonly_user`.
2. **Incomplete `ALTER DEFAULT PRIVILEGES` line**:
   You missed the `GRANT SELECT ON TABLES TO readonly_user` part.
3. **Password missing**:
   You had `PASSWORD****` — you need to provide an actual password string in single quotes.

💡 Explanation

- `GRANT USAGE ON SCHEMA` allows the user to access the schema but not the tables yet.
- `GRANT SELECT ON ALL TABLES` gives read-only access to existing tables.
- `ALTER DEFAULT PRIVILEGES` ensures that future tables will automatically grant SELECT to this user.
  ```

- `REVOKE` commands remove access to the `public` schema to ensure the user cannot access or see objects in that schema.

✅ Let me know if you'd also like a **full rollback script**, or additional security tips (like restricting functions, views, etc.).

==========

✅ To **check if a user has only read-only (SELECT) access** in Amazon Redshift (or PostgreSQL), you can run a few inspection queries. Here's a step-by-step guide:

1️⃣ Check schema privileges

```
SELECT nspname AS schema,
       usename AS user,
       has_schema_privilege(usename, nspname, 'USAGE') AS usage,
       has_schema_privilege(usename, nspname, 'CREATE') AS create
FROM pg_namespace, pg_user
WHERE usename = 'readonly_user';

```

2️⃣ Check table privileges

```
SELECT n.nspname AS schema,
       c.relname AS table,
       u.usename AS user,
       has_table_privilege(u.usename, c.oid, 'SELECT') AS select,
       has_table_privilege(u.usename, c.oid, 'INSERT') AS insert,
       has_table_privilege(u.usename, c.oid, 'UPDATE') AS update,
       has_table_privilege(u.usename, c.oid, 'DELETE') AS delete,
       has_table_privilege(u.usename, c.oid, 'REFERENCES') AS references
FROM pg_class c
JOIN pg_namespace n ON n.oid = c.relnamespace
JOIN pg_user u ON u.usename = 'readonly_user'
WHERE c.relkind = 'r'  -- only tables
  AND n.nspname NOT IN ('pg_catalog', 'information_schema');

```

3️⃣ Check default privileges on future tables

```
SELECT defaclobjtype AS object_type,
       defaclnamespace::regnamespace AS schema,
       defaclacl
FROM pg_default_acl
WHERE defaclrole = 'readonly_user';

```

💡 Explanation

- The **first query** checks whether the user can use the schema and create objects inside it.
- The **second query** checks whether the user has SELECT, INSERT, UPDATE, DELETE, REFERENCES privileges on each table.
- The **third query** checks if default privileges are set to automatically grant SELECT in future tables.

✅ Expected result for read-only user:

- Only `USAGE` on the schema = **true**.
- Only `SELECT` on tables = **true**.
- All other privileges (INSERT, UPDATE, DELETE) = **false**.

If you'd like, I can also prepare example output tables to help you easily interpret these results. Would you like me to include that?

==========

✅ Got it! You want to **grant read-only (SELECT) access** to additional schemas for your `readonly_user`.

Here's a clear step-by-step set of SQL statements to do this **safely and consistently**, similar to what you did earlier.

🔹 Example — Grant read-only access to schema: `salesanalytics`

1️⃣ Grant usage on the new schema

```
GRANT USAGE ON SCHEMA salesanalytics TO readonly_user;

```

2️⃣ Grant SELECT on all existing tables in that schema

```
GRANT SELECT ON ALL TABLES IN SCHEMA salesanalytics TO readonly_user;

```

3️⃣ Ensure future tables also grant SELECT automatically

```
ALTER DEFAULT PRIVILEGES IN SCHEMA salesanalytics
GRANT SELECT ON TABLES TO readonly_user;

```

🔹 Repeat for each new schema

If you also want to grant access to other schemas (for example, `marketingdata`, `finance`, etc.), repeat the same three steps:

```
GRANT USAGE ON SCHEMA marketingdata TO readonly_user;

GRANT SELECT ON ALL TABLES IN SCHEMA marketingdata TO readonly_user;

ALTER DEFAULT PRIVILEGES IN SCHEMA marketingdata
GRANT SELECT ON TABLES TO readonly_user;

```

⚠️ Important notes

✅ Do **not** grant any INSERT, UPDATE, or DELETE permissions.
✅ Do **not** grant CREATE privileges on schema if you want to strictly limit to read-only.
✅ Check existing privileges to avoid conflicts (let me know if you'd like ready-to-run check scripts).

✅ Let me know all the schema names — I can prepare a ready-to-copy script for you for all of them in one go!